

Word 2000 VBA

一册通

计 算 机 类 丛 书

www.BOOKOO.com.cn

Word 2000 VBA 一册通

博 库

中国 美国 台湾

前　言

Microsoft Word 2000 是微软公司 Office 2000 产品套件的一个重要组成部分。Word 一直是字处理软件行业的主流产品。作为新一代的 Word，Word 2000 不仅继承了 Word 系列原有的易用、高效的特点，而且又增加了许多激动人心的新功能。

Visual Basic for Application 是集成在整个 Office 产品中的开发语言和开发环境。在 Office 产品套件占据当今办公软件市场的情况下，掌握 VBA 来开发 Office 应用程序是十分必要的。Office 2000 中一致的开发环境和开发语言极大地减小了开发人员在构建解决方案和创建 Office 应用程序的难度。

本书由浅入深，从简单的 Word 2000 基本使用方法讲起，逐渐加深内容介绍了 VBA 语言的基本语法，最后着重讲述了 VBA 在 Word 2000 中的应用。全书共分三部分，第一部分主要介绍 Word 2000 的基本功能，为不熟悉 Word 的用户提供了一个入门的捷径；第二部分主要介绍了 VBA 的基本语法，包括数据类型、语句、函数、过程、调试等基础内容，方便用户掌握 VBA 语言；第三部分主要介绍了应用程序的高级开发，其中讲述了许多开发应用程序的高级技术，帮助用户掌握开发应用程序的方法。

本书适用的读者范围很广，既可以帮助 Word 2000 的初学者快速入门，也可以让已经掌握 Word 2000 基本操作的用户提高自己开发应用程序的能力。

在本书的创作过程中，刘敏、胡升腾、黄贝佳、张大为、邓巍巍等同志为我们提供了热情的帮助，可以说没有他们就无法完成本书的创作，在此向他们表示诚挚的谢意。

由于作者水平有限，尽管经过精心的修改，书中肯定有许多疏漏与错误之处，但发现错误本身也是对用户个人能力的锻炼，希望用户在使用本书时慎重把握。

1999 年 12 月 12 日

第一章　Word 2000 的新增功能

Word 一直是深受人们喜爱的字处理软件。随着微软公司 Office 2000 系列软件的推出，Word 2000 也作为其重要的组件之一而面世。Word 2000 与早期的 Word 版本具有很好的延续性，这使得以前使用过 Word 的用户能很容易地掌握 Word 2000。同以往的版本相比，Word 2000 的功能更加强大，它不仅能非常简便地处理诸如报告、信件等常见文档，在表单、备忘录、论文等专业文件的处理上也显得应付自如。由于 Word 2000 大大加强了网络功能，人们还可以用它作浏览器、用它制作 Web 页、用它处理电子邮件等。几乎无所不能的字处理功能及多种附加功能，简便易学的操作以及友好直观的界面，所有这一切，使得 Word 2000 成为大多数人首选的字处理软件。

作为新一代的 Word 产品，Word 2000 不仅继承了 Word 系列原有的易用、高效的特点，而且又增加了许多激动人心的新功能。本章将对此进行简要的介绍。部分内容还将在本书其他章节中详细介绍。

1.1　即点即输功能

Word 2000 新增了“即点即输”的功能，利用这一功能可以在文档的空白区域快速插入文字、图形、表格或其他项目。

所谓“即点即输”，就是指当在文档的空白处双击鼠标时，Word 2000能够自动将鼠标双击的位置变为当前输入位置，并且自动设置输入的格式和样式。这样做带来的方便之处是显而易见的，例如，如果要在文档末尾的下面输入图形，就可以利用“即点即输”功能，直接用鼠标来控制图形的位置，而不必先按“Enter”键添加空行，再利用空格键将光标移动到相应位置。

使用“即点即输”功能时，还可以控制输入的格式和样式。将鼠标指针移动到选定的输入位置后，指针的形状即表明了输入的格式。例如，如果鼠标指针的形状为 I，表示要输入的内容是居中放置的。使用“即点即输”功能输入的文本有默认的样式，可以通过打开菜单栏中的“工具”菜单，并选择其中的“选项”命令来修改。

✍ “即点即输”功能只能在“页面”视图和“Web 版式”视图下使用（视图的概念将在第二章《文档的格式》中介绍）。此外，不能在以下区域内使用“即点即输”功能：多栏、项目符号和编号列表、浮动对象旁边、具有上下型文字环绕方式的图片的左边或右边、缩进的左边或右边。

1.2 神通广大的 Office 助手

同以往的 Word 版本一样，Word 2000 的帮助功能十分强大。Word 2000 提供了十个 Office 助手，它们比以往更加栩栩如生，更加亲切可

人。用户可以选择自己喜欢的助手形象，例如可以让“智多星”爱因斯坦来出谋划策，如图 1-1 所示。这些 Office 助手神通广大，能提供用户所需要的各种帮助。它们不仅可以回答用户的问题，实时地给予提示，还能够提供有关 Office 程序的各种性能的帮助信息。

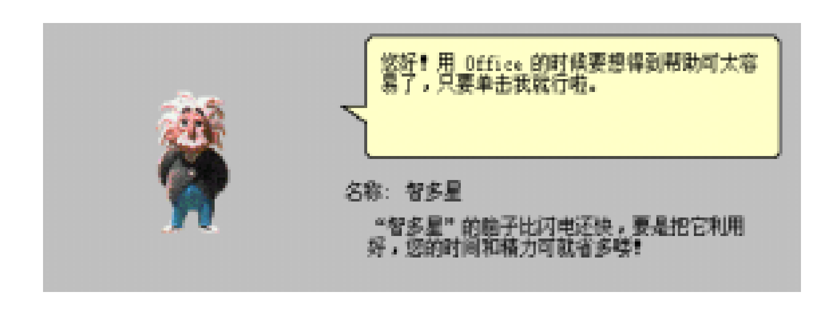


图 1-1　Word 2000 中的 Office 助手

✍ 如果不喜欢 Office 助手，可以将其隐藏或者删除。值得注意的是：Office 助手是所有 Office 程序所共享的，因此，对 Office 助手的任何修改都会影响到所有的 Office 程序。

1.3　别具匠心的菜单和工具栏

Word 2000 的菜单和工具栏的设计别具匠心。如果用户是第一次使用 Word 2000，打开菜单时可能会大吃一惊：与 Word 97 相比，菜单中少了很多命令。这就是 Word 2000 独特的个性化菜单设计。在一般情况下，Word 2000 的菜单只显示最常用的命令，这样使菜单显得更加简洁，有助于提高使用 Word 的效率。同时，用户很容易对菜单进行定制。只要用户使用了某一个命令，下一次这个命令就会出现在个

性化的菜单内。如果想要展开一个菜单内的所有命令，可以单击菜单最下方的或者双击菜单栏中的菜单名。例如，单击菜单栏中的“工具”菜单名可以打开“工具”菜单，而双击“工具”菜单名即可展开“工具”菜单的所有命令，如图 1-2 所示。

在 Word 2000 的初始设置中，工具栏只占一行，以便留出更多的屏幕空间用于编辑文件。如果用户使用了工具栏中的某个按钮，则该按钮将被添至屏幕上的个性化工具栏中。用户还可根据自己的需要自定义工具栏。



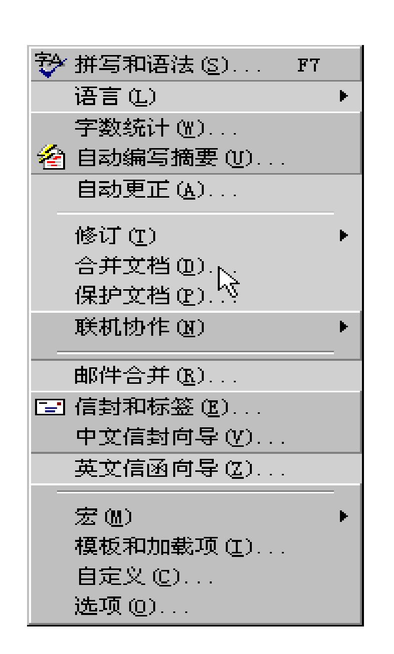


图 1-2　个性化的“工具”菜单和展开的“工具”菜单

1.4　强大的剪贴板工具

Word 2000 的剪贴功能得到了很大的增强。通过使用如图 1-3 所示

的剪贴板，用户可以在任意的程序中收集（即复制）多个对象，并在 Word 2000 或者其他的 Office 程序中进行粘贴。Word 2000 的剪贴板中最多可以容纳 12 个对象，粘贴时可以选择粘贴其中的任意一个或全部粘贴。



图 1-3　Word 2000 的剪贴板

要想收集并粘贴多个对象，必须保证 Word 2000 的剪贴板是打开的。如果 Word 2000 没有显示剪贴板，可以在菜单栏、“常用”工具栏或“格式”工具栏的任意位置单击鼠标右键，并从弹出的快捷菜单中选择 “剪贴板”命令。此外，在同一个程序中连续收集两个对象也可以自动打开“剪贴板”。

1.5　全新的打开和保存文档的对话框

Word 2000 改进了打开和保存文档的对话框。

全新的“打开”对话框如图 1-4 所示。可以看到，与以往版本的

Word 相比，Word 2000 中的“打开”对话框新定义了五个查找文件的起始位置：“历史”、“My Documents”、“桌面”、“收藏夹”以及“Web 文件夹”，这样就使用户能更加快捷地查找到所需文件。同时，可以利用“打开”按钮的下拉菜单选择打开文件的方式（如以只读方式打开，打开文档的副本等等）。此外，在对话框的右上方增添了一个工具栏，可以方便地对文档进行查找、删除、查看属性等操作，更加方便了对文档的处理。

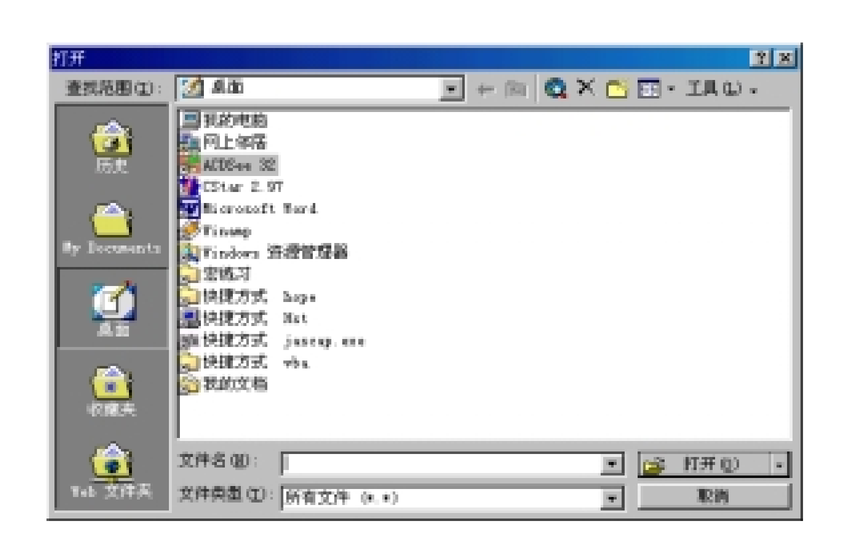


图 1-4　“打开”对话框

用于保存文档的“另存为”对话框与“打开”对话框类似。

1.6　增强了宏命令的安全性

Word 2000 增强了对使用文档中宏命令的安全性检查。打开菜单栏中的“工具”菜单，并选择“宏”子菜单中的“安全性”命令，将出

现如图 1-5 所示的“安全性”对话框，可以从中更改宏病毒防护功能的安全级。各安全级的防护功能如下：

1．“高”安全级

选中“高”安全级后，Word 2000 将只运行文件中具有数字签名、并且来自经用户确认为可以信任的来源的宏。在信任某一来源之前，应当确认该来源可靠并且会在签发宏之前使用病毒扫描程序检查病毒。没有数字签名的宏将被自动禁用，Word 2000 将打开文档，并且不发出任何警告。

2．“中”安全级

选中“中”安全级后，如果 Word 2000 遇到的宏的来源不在可靠来源列表中，则会显示警告信息，用户可以选择打开文档时是启用还是禁用宏。如果该文档可能带有病毒，则应选择禁用宏。

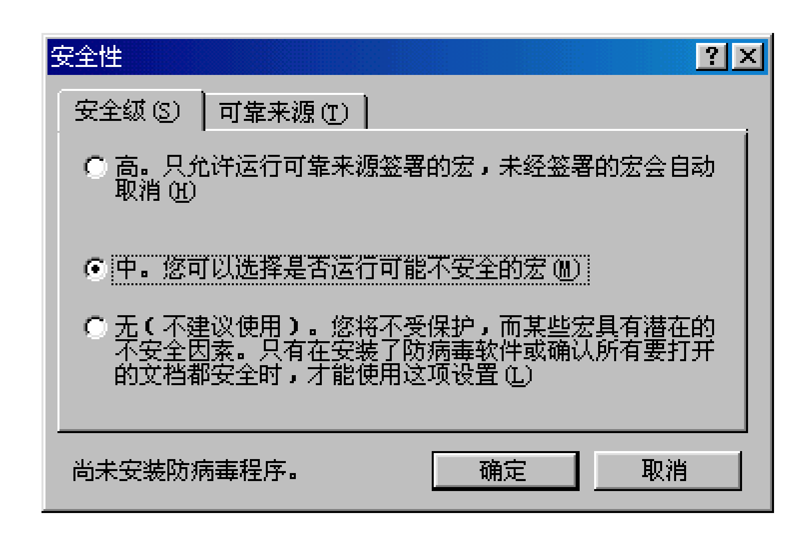


图 1-5　“安全性”对话框

3. “无”安全级

如果用户确信所有要打开的文档和加载项都是安全的，可以选中“无”安全级。选中“无”安全级后，Word 2000 将关闭宏病毒防护功能。在此安全级下，打开文档时将自动启用文档中所有的宏。

1.7 更强大的网络功能

与 Word 的以往版本相比，Word 2000 的网络功能更加强大，很好地适应了字处理软件网络化的要求。

Word 2000 支持将 Word 文档保存为 Web 页的形式，用户可以轻松地在 Word 2000 中制作自己的网页。

通过使用常用的标记，例如表格、字体和背景声音，Word 2000 为创建网页提供所见即所得的支持。用户无需掌握专门用于创建 Web 页的 HTML 语言就可以创建 Web 页（如果要查看 Word 2000 中创建的 Web 页的 HTML 代码，可以打开菜单栏中的“视图”菜单，并选择其中的“HTML 源文件”命令）。

在 Word 2000 的 Web 版式视图中，用户无须离开 Word 即可查看 Web 页在 Web 浏览器中的效果。预览 Web 页时，打开菜单栏中的“文件”菜单，并选择其中的“Web 页预览”命令，此时系统将启动 Windows 默认的浏览器（例如 Internet Explorer），同时在该浏览器中打开 Web 页。

打开菜单栏中的“工具”菜单，并选择其中的“选项”命令，将出现一个“选项”对话框，单击“常规”选项卡中的“Web 选项”按钮将出现如图1-6所示的“Web 选项”对话框。在这个对话框中可以快速更改Office 2000 生成 Web 页以及设置其格式的方式。例如，可以自定义 Web 页的图形和其他支持文件的保存位置，也可指定 Web 页中所使用的图形的保存格式。

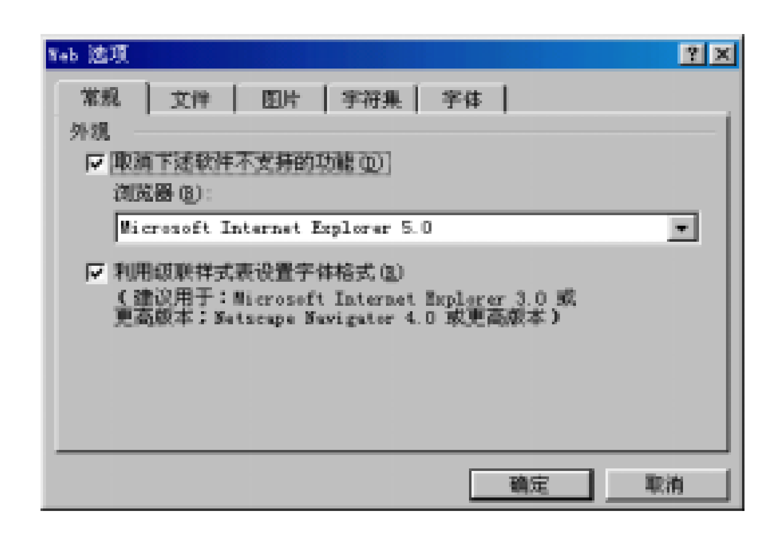


图 1-6　“Web 选项”对话框

在 Word 2000 中编辑 Web 页时，可以使用脚本，使创建 Web 网页变得更加轻而易举。Office 2000 允许用户使用 Visual Studio 开发环境，在任意 Office 程序中创建基于脚本和 HTML 的客户端解决方案。

在 Word 2000 中创建 Web 页后，所有的支持该 Web 页的文件都保存在与该文件同名的目录中。如果用户将文件保存在新的位置中，Word 2000 将检查链接并修复那些无法工作的链接。

Word 2000 新增了许多关于电子邮件的功能。例如，如果将文档新建为电子邮件，就可以用 Word 2000 作为电子邮件编辑器。Word 2000 允许“邮件”合并到“电子邮件”，指定 E-mail 地址以后直接进行发送。也可直接在 Word 2000 中通过电子邮件发送文件的副本。

打开菜单栏中的“工具”菜单，并选择其中的“选项”命令，将出现一个“选项”对话框，单击“常规”选项卡中的“电子邮件选项”按钮将出现如图 1-7 所示的“电子邮件选项”对话框。在这个对话框中可以对电子邮件的格式进行设置。

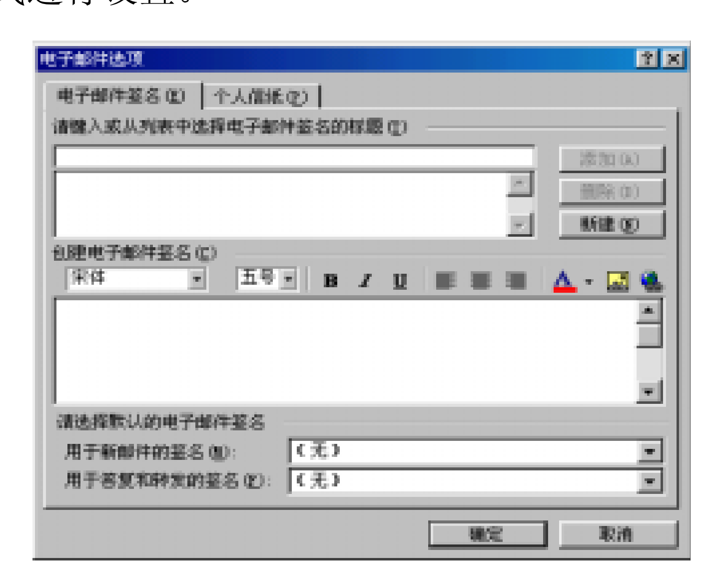


图 1-7　“电子邮件选项”对话框

1.8　中文拼写和语法检查

Word 2000 提供了经过改进的拼写和语法检查工具。例如，现在的拼写检查工具能够识别更多的人名、组织和公司的名称、城市和国家

及地区、Internet 和文件地址等信息。Word 2000 的语法检查工具能标记错误，并且能够对错误进行自动更正。

此外，Word 2000 还能够对中文进行文字的拼写和语法检查，这对中文用户无疑是个很大的喜讯。

1.9　简体中文和繁体中文的转换

Word 2000 支持简体中文和繁体中文之间的转换。进行繁体和简体转换时，可以使用“常用”工具栏上的“中文繁简转换”按钮繁来进行。也可打开菜单栏中的“工具”菜单，并选择“语言”子菜单中的“中文繁简转换”命令。在进行繁简转换时，如果选定了文字，则对选定的文字进行转换；如果没有选定任何文字，则对整篇文档进行转换。

第二章　文档制作进阶

作为字处理软件，Word 2000 最基本的功能之一就是用来制作各种各样的文档。相信读者对于 Word 文档的制作方法有或多或少的了解，也熟悉常用的操作，例如创建文档、打开文档、编辑文档（包括选定、键入、删除、剪切、复制、粘贴、移动、撤消、恢复、重复、查找、替换、定位等）、保存文档、打印文档，以及进行字体格式和段落格式的编排等。这些内容也可在任何一本 Word 2000 的基础教程里找到。本章将在此基础上，更加深入地介绍一些文档制作方面的知识。通过本章的学习，读者可以更快、更好地制作出精美的文档。

2.1　文档视图

文档视图是指编辑文档时文档的显示方式。由于文档的编辑以及对文档格式的处理总是在一定的视图状态下进行的，因此对文档视图的介绍是很有必要的。Word 2000 提供了六种视图形式，分别是普通视图、页面视图、大纲视图、Web 版式视图、文档结构图以及打印预览视图。与 Word 97 相比，Word 2000 中新增加了两种视图：Web 版式视图和文档结构图。本节将重点介绍这两种视图。

2.1.1　视图切换概述

文档从一种显示方式转换为另一种显示方式叫做视图切换。各种视图相互切换的时候，文档本身的内容是不会发生变化的。

在 Word 2000 编辑窗口的底部有一个水平滚动条，其左端有四个视图按钮，从左至右依次为普通视图按钮，Web 版式视图按钮，页面视图按钮，以及大纲视图按钮。单击某个按钮可以选择相应的文档视图方式，从而实现这四种视图状态之间的切换。

也可利用“视图”菜单中的命令来切换视图。打开菜单栏中的“视图”菜单，可以看到其中包括了“普通”、“Web 版式”、“页面”、“大纲”以及“文档结构图”等命令，单击这些命令可以切换到相应的视图状态。

打开菜单栏中的“文件”菜单，并选择其中的“打印预览”命令，或者直接单击“常用”工具栏中的“打印预览”按钮，即可切换到打印预览视图状态。

文档一定是而且只能是在普通视图状态、页面视图状态、大纲视图状态、Web 版式视图状态以及打印预览视图状态中任意一种状态下显示。文档结构图只能与普通视图、页面视图、大纲视图以及 Web 版式视图之一同时显示。在打印预览视图状态下不能显示文档结构图。

2.1.2　Web 版式视图

Word 2000 中新增了 Web 版式视图。在 Web 版式视图中，可以创建能显示在屏幕上的 Web 页或文档。Web 版式视图状态如图 2-1 所示。如果用户希望使用 Word 2000 来编辑网页，Web 版式视图当然是首选的视图形式，因为在 Web 版式视图状态下所显示的网页与在浏览器中显示出来的网页几乎完全相同。

在 Web 版式视图中，可以看到能够根据窗口大小而自动换行显示的文本以及文本背景，且图形位置与在 Web 浏览器中浏览时的位置一致。

✍ 打开菜单栏中的“文件”菜单，并选择其中的“Web 页预览”命令，可以打开 Windows 默认的 Web 浏览器预览文档在 Web 浏览器中的外观。

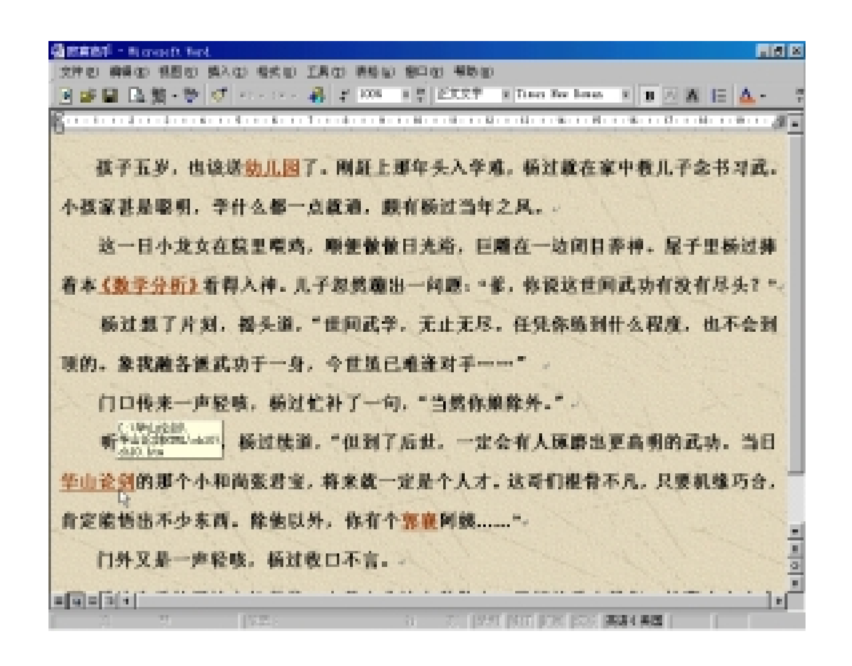


图 2-1　Web 版式视图状态

2.1.3　文档结构图

文档结构图与大纲视图有类似的地方，可以帮助用户更好地查看文档结构或者更方便地到达文档中特定位置。文档视图是文本区左侧的一个窗口，如图 2-2 所示，它是与普通视图、页面视图、大纲视图以及 Web 版式视图之一同时显示的（在打印预览视图状态下不能显示文档结构图）。

文档结构图实际上是文档的标题栏。单击这些标题时，右边窗口中的文档就会显示到标题所对应的位置处。单击文档结构图中标题前的加、减号，分别可以展开和折叠将该标题下的小标题。

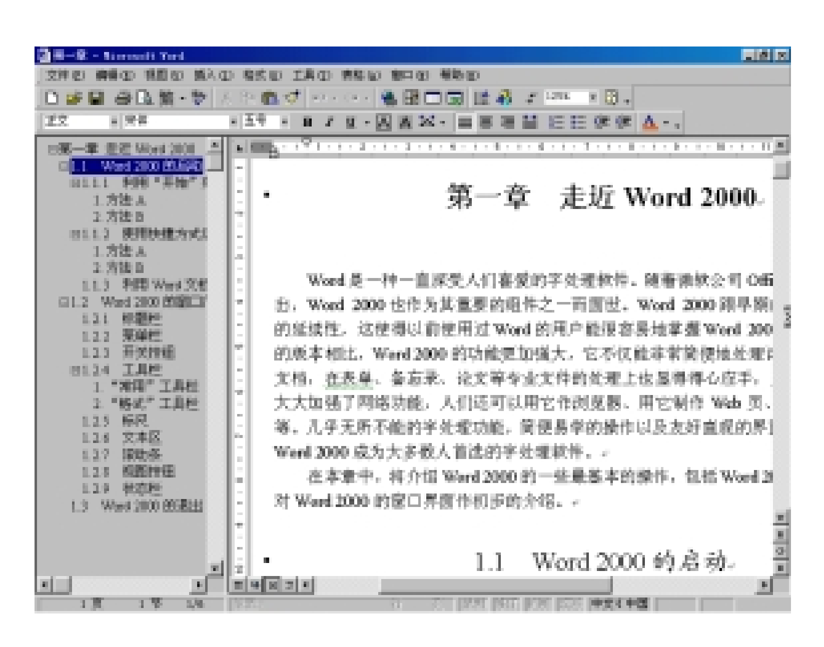


图 2-2　文档结构图

2.2　样式

在文档的编辑过程中，可以说处处都要与“格式”打交道。通过字符格式、段落格式的设置，可以使文档的外观更加友好，层次更加分明，重点更加突出。有时，文档需要进行多处格式化处理，而其中很多部分的格式又是基本相同的。在遇到诸如此类的情况时，常用的设置格式的方法就显得过于机械和烦琐了。为此，Word 2000 提供了样式和模板功能，可以快捷方便地对文档进行自动格式化。在本节及下一节中，将分别对样式和模板进行介绍。

样式是具有特定名称的一系列格式特征的组合，换句话说，样式是格式的集合。由于应用样式时，只需执行一步操作就相当于执行一系列的格式化操作，因此利用样式可以快速地改变文本格式。

Word 2000 的样式包括段落样式和字符样式两种。其中段落样式用于设置段落外观的格式，例如字体设置、对齐方式、缩进格式、制表位、行间距等；而字符样式用于设置段落内选定文字的格式，例如文字的字体、字号、字形等。即使某段落已整体应用了某种段落样式，该段中的字符仍可以有自己的样式。

✍ 应用样式、基于已有格式新建样式等操作都需要先选定相应的文本，因此本节中将屡次提到“选定文本”之类的词。我们约定，在本节当中，对于段落样式，选定文本是指选

定相应段落（或者至少选定各个段落的一部分），或者将光标移到相应的段落中（对于单个段落而言）；对于字符样式，选定文本则是指选定相应的文字。

2.2.1　应用样式

首先来看一看如何应用样式来设置文档格式。

基于 Word 2000 的默认模板（Normal 模板）所创建的文档提供了“正文”、“默认段落字体”、“标题 1”、“标题 2”、“标题 3”等基本样式。当要改变段落或文字的格式时，可以对其应用相应的样式。如果所需的样式尚未建立，则须新建样式之后才能应用。

应用样式通常有以下几种途径：使用工具栏中的“样式”框，使用“样式”对话框以及使用快捷键。下面将分别介绍。

1. 使用工具栏中的“样式”框应用样式

使用“格式”工具栏中的“样式”框可以很方便地对文档应用已有的可用样式。

首先选定要应用样式的文本，然后单击“格式”工具栏中“样式”框右端的下拉箭头，将出现一个下拉列表。其中列出了当前文档所使用的一些样式；如果按住“Shift”键单击下拉箭头，则下拉列表中将列出当前文档所有可用的样式。在样式列表中选中所需的样式后，单击鼠标左键即可。

2. 使用“样式”对话框应用样式

使用“样式”对话框对文档应用样式时，首先选定要应用样式的文本，然后打开菜单栏中的“格式”菜单，并选择其中的“样式”命令，此时将出现如图 2-3 所示的“样式”对话框。从中可以应用样式、查看样式说明并预览应用样式后的段落及字符效果。



图 2-3　“样式”对话框

单击“列表”栏右端的下拉箭头，可以看到下拉列表中包括了“正在使用的样式”、“所有样式”和“用户定义的样式”等三个选项。选中某一选项后，在“样式”框中将列出相应的样式。如果样式名前有“a”，说明这是字符样式；如果样式名前有“↵”，说明这是段落样式。从中选择所需的样式后，在对话框右边的预览框里可以预览样式效果，

在“说明”栏里将出现对该样式的具体说明。确信是所需样式后，单击“应用”按钮即可。

3. 使用快捷键应用样式

对于几种常用的样式，Word 2000 提供了快捷键，以便能够快速方便地对文档应用样式。选定要应用样式的文本后，按下快捷键“Ctrl+Shift+N”可以应用“正文”样式，按下快捷键“Ctrl+Alt+1”可以应用“标题 1”样式，按下快捷键“Ctrl+Alt+2”可以应用“标题 2”样式，按下快捷键“Ctrl+Alt+3”可以应用“标题 3”样式，按下快捷键“Ctrl+Shift+L”可以应用“列表项目符号”样式。用户也可以自己为样式定义快捷键，定义快捷键的方法将在稍后“新建样式”中介绍。

2.2.2 新建样式

以上介绍了如何应用已有的样式。这些样式可能是 Word 2000 自带的（样式的种类由当前使用的模板决定），也可能是用户自己定义的。在实际应用中，Word 2000 所提供的样式往往不能满足用户的具体需要，新建样式就是很必要的了。

新建样式通常有两种途径：使用工具栏中的“样式”框以及使用“新建样式”对话框。以下将分别进行介绍。

1. 使用工具栏中的“样式”框新建样式

对于已经格式化了的文本，使用“格式”工具栏中的“样式”框可以基于现有格式快速地新建段落样式。

首先选定作为新建段落样式样本的文本，然后单击“样式”框，删除原有的样式名并在框中键入新的样式名后，按回车键即可。新建的样式与所选文本的样式相同，并可用于对文档的其他部分进行格式化。

这种方法不能用以新建字符样式。

2. 使用“新建样式”对话框新建样式

按照前面的方法打开图 2-3 所示“样式”对话框后，单击“新建”按钮，将出现如图 2-4 所示的“新建样式”对话框。使用“新建样式”对话框新建样式，不仅可以新建段落样式，也能够新建字符样式，还具有预览样式效果、为新建的样式指定快捷键，设置段落样式的其他格式特征等功能。

在“名称”框内键入新建样式的样式名。一般来说，样式名应当简洁直观，最好能对该样式作概括性地描述，例如“图-居中”、“节首段，小四黑宋体”、“标题 5，红，右对齐”等。新建样式不能与已有样式重名。

单击“样式类型”框右端的下拉箭头将出现一个下拉列表框，其中包括“段落”和“字符”两个选项，分别表示新建的样式为段落样

式或者字符样式。选中所需类型即可。

可以从“基准样式”框的下拉列表中为新建样式选择基准样式。如果不需要基准样式，则选择“无样式”。当要新建的样式与某种已有的样式类似时，可以选择该已有样式作基准样式并在此基础上进行修改，这样可以简化建立新样式的过程。

如果新建的是段落样式，那么在“后续段落样式”框的下拉列表中可以选定应用新建样式的段落的后续段落的样式。

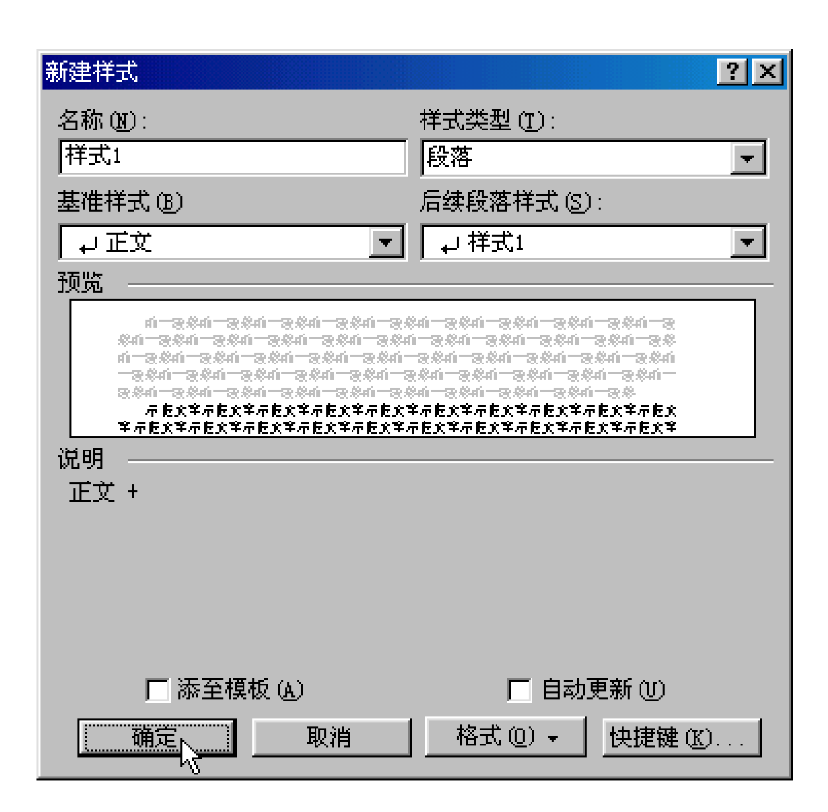


图 2-4　“新建样式”对话框

单击“新建样式”对话框中的“格式”按钮将弹出如图 2-5 所示的下拉菜单，从中选择要指定的属性后就会出现相应的格式对话框，此

时可以对样式的格式进行设置。设置完一项属性后，可单击“格式”按钮继续选择其他属性进行设置。直到将新建样式的各种格式设置完成。

字体(F)...
段落(P)...
制表位(T)...
边框(B)...
语言(L)...
图文框(M)...
编号(N)...

图 2-5　“格式”下拉菜单

如果想为新建的样式指定快捷键，可以单击“新建样式”对话框中的“快捷键”按钮，此时将出现如图 2-6 所示的“自定义键盘”对话框。将光标移到“请按新快捷键”框中，然后按下要指定的快捷键，在“目前指定到”栏确认其尚未被指定为其他操作的快捷键后，单击“指定”按钮即可。单击“关闭”按钮可返回“新建样式”对话框。

完成对新建样式的格式设置之后，单击“新建样式”对话框的“确定”按钮即可。

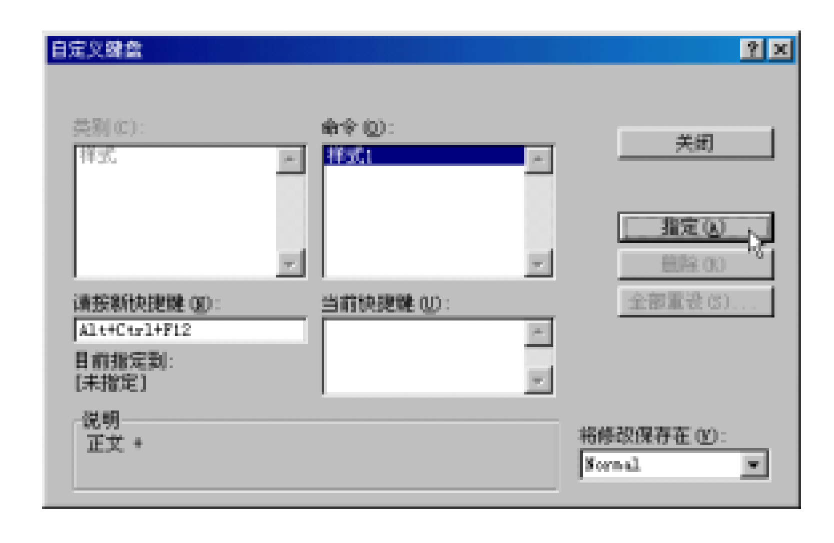


图 2-6 “自定义键盘”对话框

2.2.3 修改样式

有时候，用户想要对具有某种样式的所有文本的格式都进行相同的改变，此时没有必要对所有这些文本都一一重设置格式，也没有必要先新建一种样式，然后对所有这些文本应用新建的样式，直接对原有的样式作适当的修改是最简单的作法。

修改样式通常有以下两种途径：使用工具栏中的“样式”框以及使用“更改样式”对话框。下面将分别介绍。

1. 使用工具栏中的“样式”框修改样式

使用工具栏中的“样式”框修改样式的一个好处是，这种修改可以直接在页面模式下进行，从而用户可以直观地看到对文本作样式修改后的效果，并作出相应的调整。

首先对要修改样式的文本作段落格式和字符格式的修改，修改完

成之后，单击“样式”框右端的下拉箭头并从下拉列表中选择要修改样式的样式名，此时将出现如图 2-7 所示的“更改样式”对话框。选定“更新样式，以反映最近所做修改”后，单击“确定”按钮即可修改样式。修改后的样式与修改过的文本的样式相同。如果用户想将修改后的样式应用于文档中所有具有修改前样式的文本，则应选中“从现在开始自动更新样式”，如果想取消样式修改，则选择“将样式的格式重新应用于所选内容”。

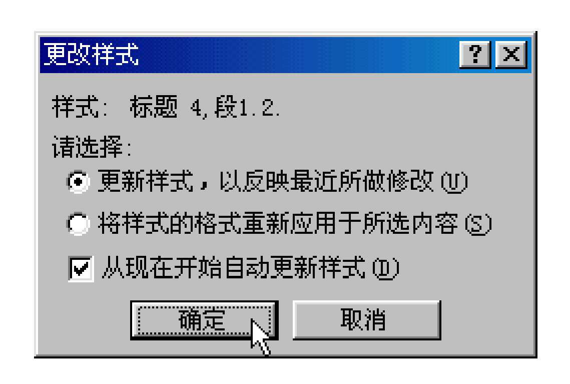


图 2-7 “更改样式”对话框

2. 使用“更改样式”对话框修改样式

按照前面的方法打开图 2-3 所示“样式”对话框并从“样式”框中选中要更改的样式后，单击“更改”按钮，将出现如图 2-8 所示的“更改样式”对话框。这个对话框与图 2-4 所示的“新建样式”对话框从外观到用法上都非常类似，因此这里只作简单介绍。

从“更改样式”对话框可以看到，样式类型是不能改变的，但是可以从相应的下拉列表中修改基准样式和后续段落样式，也可以单击

“快捷键”按钮调出图 2-6 所示的“自定义键盘”对话框指定或修改快捷键。

修改样式各项属性格式的方法与新建样式时完全相同，只要单击“格式”按钮并从其下拉菜单中选择要修改的属性进行格式设置即可。

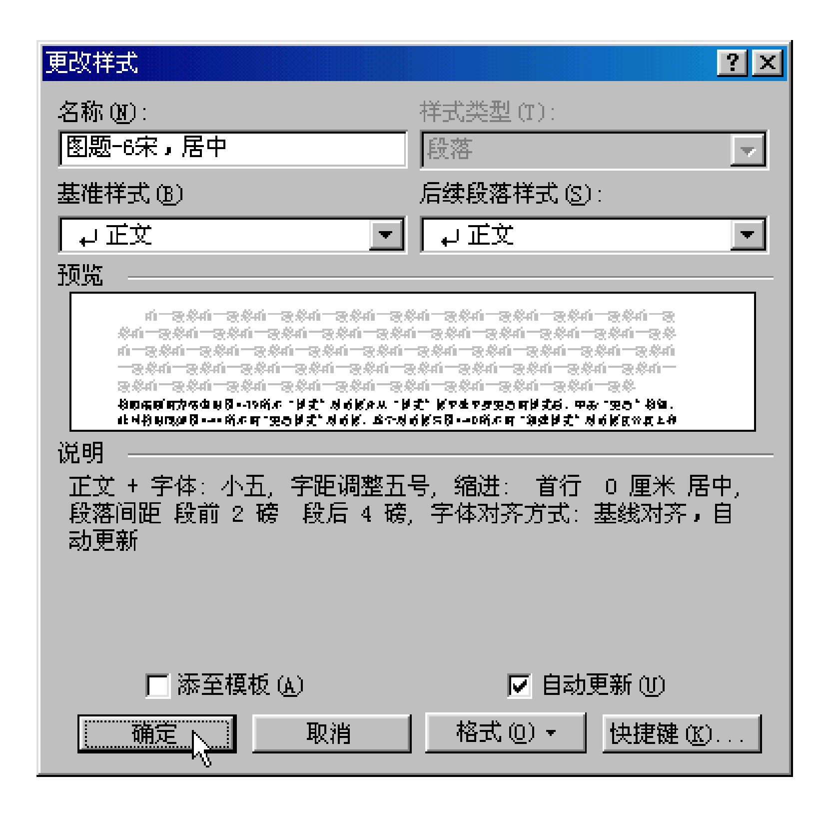


图 2-8　“更改样式”对话框

当对样式的修改全部完成后，单击“确定”按钮即可。

✍ 应当注意，如果对文档基准样式的某种格式设置进行了修改，则文档中基于此基准样式的所有样式都将相应被修改。

2.2.4　删除样式

用户可以删除自定义的样式。

删除样式的方法很简单，首先按照前面的方法打开图 2-3 所示的“样式”对话框，然后从中选中要删除的样式，单击“删除”按钮或者按快捷键“Del”将出现确认删除的对话框，单击“确定”按钮即可。

- 如果单击了“列表”框中的“正在使用的样式”选项，则可以删除特定文档中的某些内置样式（正文样式和内置标题样式除外）。Word 2000 将对所有具有这些样式的段落应用“正文”样式。但通过单击“列表”框中的“所有样式”选项仍可重新获得删除的内置样式。
- 如果用户删除了自定义的段落样式，Word 2000 将对所有具有此样式的段落应用“正文”样式。自定义的样式删除后不能重新获得。

2.3 模板

模板是一种只读文档，其扩展名为.Dot。它包含了各种样式以及其他一些文档的基本元素，例如工具栏、菜单、宏、自动图文集词条、快捷键、页面布局等。Word 2000 中的任何文档都是在一定的模板上建立的。模板是大量文档所具有的共同设置、指令以及文档内容的集合，文档的模板决定了文档的基本结构和文档设置等特征。

Word 2000 中的模板分为共用模板和文档模板两种基本类型。其中文档模板所含设置仅适用于基于该模板而创建的文档，而共用模板（包

括 Normal 模板）所含设置可以适用于所有的文档。例如，用户基于“典雅型报告”模板新建了文档，则该文档既能使用包含在“典雅型报告”模板中的设置，也能使用包含在任何共用模板中的设置。

2.3.1 应用模板

1. 用模板创建文档

事实上，任何文档都是基于特定模板而创建的。如果创建文档时用户没有指定模板，则 Word 2000 将默认模板为 Normal 模板（Normal.Doc）。

打开菜单栏中的“文件”菜单，并选择其中的“新建”命令，将出现如图 2-9 所示的“新建”对话框。我们知道，此时可以创建新文档。事实上，此时对话框中各图标代表的正是不同的模板。例如单击“常用”标签打开“常用”选项卡可以看到一个“空白文档”图标，它代表的就是 Normal 文档。



图 2-9 “新建”对话框

从“新建”对话框中可以看到，Word 2000 提供了多种模板类型，例如“常用”类、“报告”类、“备忘录”类、“信函和传真”类等，可以单击相应的标签打开各选项卡，图 2-9 显示的就是“出版物”选项卡。在各个模板类型中又包括了多种具体的模板，例如“出版物”类模板中就包括了“典雅型通讯”模板、“论文”模板、“目录”模板、“手册”模板等。选定所需模板后，单击“确定”按钮，即可创建基于所选模板的新文档。

2. 加载模板

如前所述，基于某特定模板创建的文档可以使用保存在该模板中以及保存在共用模板（包括 Normal 模板）中的设置。然而有的时候，用户可能希望使用保存在其他模板中的设置，此时可以加载所需的模板。被加载的模板将成为共用模板从而能被所有文档使用。

打开菜单栏中的“工具”菜单，并选择其中的“模板和加载项”命令，将出现如图 2-10 所示的“模板和加载项”对话框。在“共用模板及加载项”框中选中要加载的模板前的复选框后，单击“确定”按钮即可。如果框内未列出所需的模板，可先单击“添加”按钮，并从出现的“添加模板”对话框中找到所需模板，然后单击“确定”按钮将其添加到“共用模板及加载项”框中以便进行选定。



图 2-10　“模板和加载项”对话框

如果用户退出了 Word 2000，重新启动 Word 后，加载的模板或加载项并不会自动重新加载。如果需要在启动 Word 2000 时自动加载模板或加载项，必须将该模板或加载项存放在启动文件夹中。

- 打开菜单栏中的“工具”菜单，并选择其中的“选项”命令，将出现“选项”对话框，单击该对话框的“文件位置”标签切换到“文件位置”选项卡，从中可以查看“启动”文件夹的路径。

将不常用的模板卸载有助于节省内存并能提高 Word 2000 的运行速度。前面已经提到，退出 Word 2000 时，加载的模板将自动卸载，重新启动 Word 后也不会自动加载（除非将其存放在启动文件夹中）。如果要卸载一个模板但仍希望将其保留在“共用模板及加载项”框中，

只需清除该项名称前的复选框即可。如果要卸载一个模板并将其从“共用模板及加载项”框中删除，则须在框内单击此项，然后单击“删除”按钮（所选模板存储于启动文件夹中时无法删除）。

3. 更改文档模板

在图 2-10 所示的“模板和加载项”对话框中还可以更改文档模板。

在“模板和加载项”的“文档模板”框中显示了文档当前所使用的模板名称。更改模板时，单击“选用”按钮将出现如图 2-11 所示的“选用模板”对话框，从中选定一个模板后，单击“打开”按钮返回“模板和加载项”对话框，此时“文档模板”框中显示的是新选定的模板。再单击“确定”按钮即可更改文档模板。

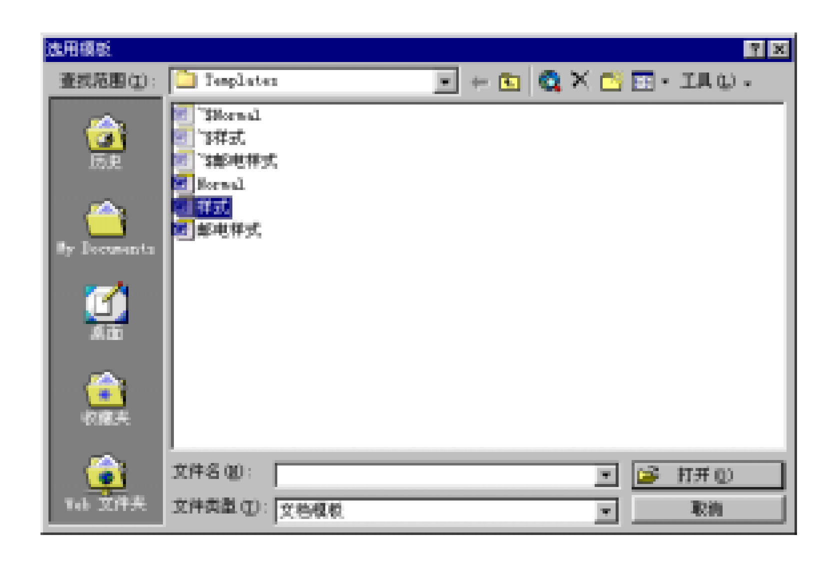


图 2-11　“选用模板”对话框

更改文档模板与前面介绍的加载模板的区别在于，更改文档模板后，原有模板将不再有效；而加载模板时，文档所基于的模板并不会

改变。

- 文档的模板更改以后，原有文档的内容并不会受到影响。如果想用现在使用的模板中的设置来更新原有文档的内容，则须在“模板和加载项”对话框中选中“自动更新文档样式”复选框。

2.3.2 修改模板

有的时候，用户需要更改模板内部的某些设置。Word 2000 允许对模板进行修改。

首先应当打开要修改的模板。打开菜单栏中的“文件”菜单，并选择其中的“打开”命令，将出现如图 2-12 所示的“打开”对话框。在“文档类型”框的下拉列表中选中“文档模板”并找到要进行修改的模板后，单击“打开”按钮即可。

- 模板的存放路径通常为：“C:\Windows_文件夹\Application Data\Microsoft\ Templates”或“C:\Windows_文件夹\Profiles\用户名\Application Data\Microsoft\ Templates”。

打开模板后就可以对其进行修改了。可以修改模板中的文本和图形、样式、格式、宏、自动图文集词条、工具栏、菜单设置和快捷键等设置。修改模板的方法与修改普通文档的方法完全相同。例如在“格式”工具栏或相应的对话框里对段落格式和字符格式进行修改，在“样

式”对话框里对样式进行修改，重新调整页面设置，重新录制宏、在模板中添加文本和图形（添加的内容将出现在所有基于该模板的新文档中）等等。

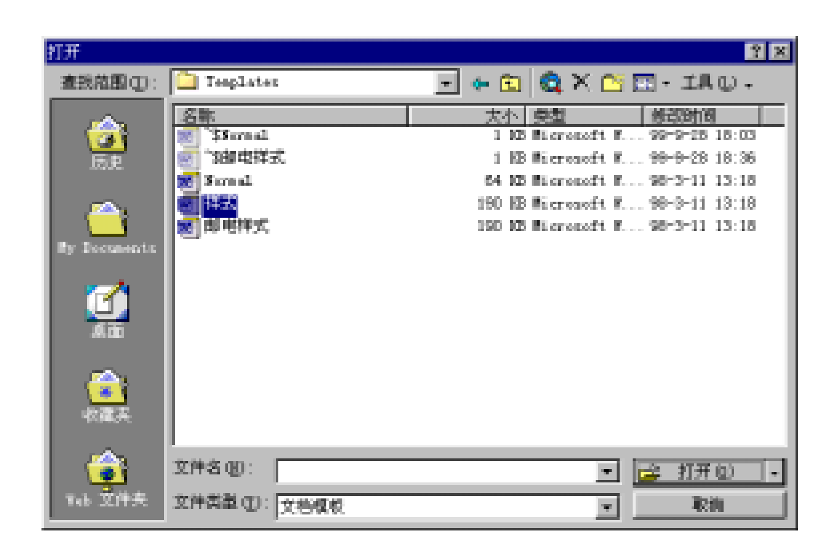


图 2-12　“打开”对话框

修改完模板后，单击“文件”菜单中的“保存”命令或“常用”工具栏中的“保存”按钮即可。也可按快捷键“Ctrl+S”将改动存盘。

✍ 模板被修改之后，基于此模板的已有文档的内容并不会受到影响。如果想用修改后的模板中的设置来更新这些文档的内容，则须在打开文档前，在图 2-10 所示的“模板和加载项”对话框中选中“自动更新文档样式”复选框。这样当再次打开已有文档时，Word 2000 将更新修改过的样式。

2.3.3　新建模板

尽管 Word 2000 提供了多种模板供用户选用，但是在很多情况下，

用户并不能从 Word 2000 自带的模板中找到一个非常合适的模板来创建自己的文档。在这种时候，尤其是需要创建多个结构类似的文档的时候，根据需要来新建一个模板就很有必要了；另一方面，这也是一种事半功倍的作法。

1. 根据已有模板新建模板

如果要新建的模板与某个已有模板很相似，则可在该以有模板的基础上对其进行加工来新建模板。

与新建文档类似，新建模板时，打开菜单栏中的“文件”菜单，并选择其中的“新建”命令，将出现一个“新建”对话框，在相应的选项卡中选中与要新建的模板相似的模板，并且在“新建”栏里选中“模板”选项后单击“确定”按钮即可。

例如，要在“现代型信函”基础上新建一个模板，则在“新建”对话框里选中“现代型信函”模板，并在“新建”栏里选中“模板”选项，如图 2-13 所示。单击“确定”按钮后将出现如图 2-14 所示的编辑界面。注意此时标题栏中显示的是“模板”而不是“文档”。在此可以根据要新建的模板的要求对原有的模板进行修改。

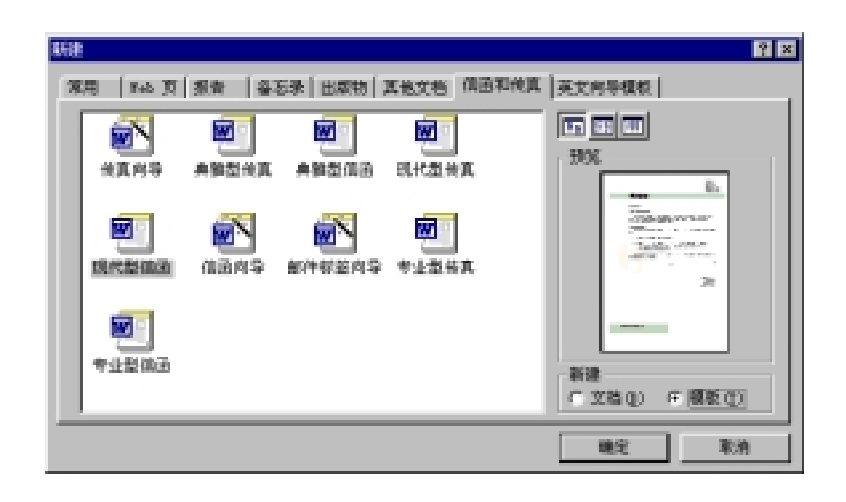


图 2-13　“新建”对话框

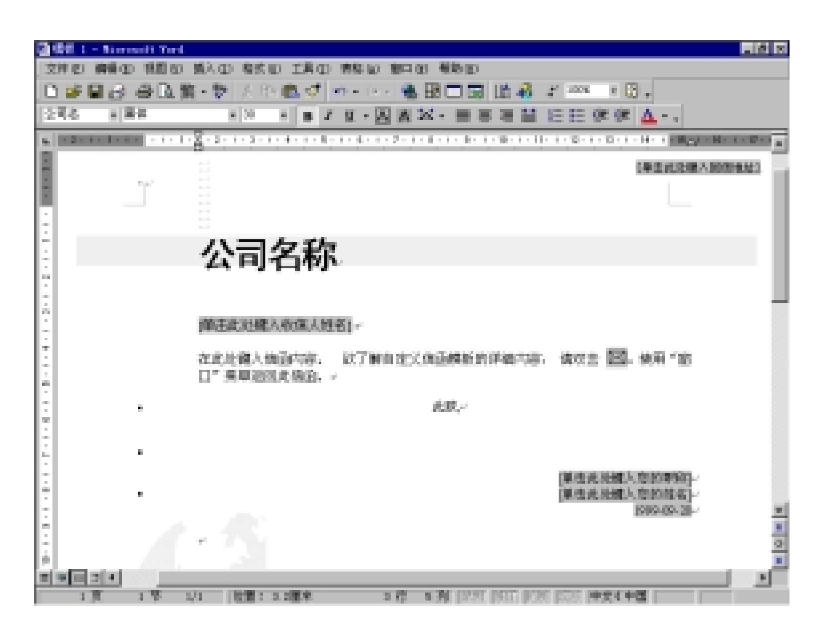


图 2-14　根据已有模板新建模板

对原有模板的内容和设置的修改完成之后，就可以将其存为新模板了。打开菜单栏中的“文件”菜单，并选择其中的“另存为”命令，将出现如图 2-15 所示的“另存为”对话框。在“保存位置”框中指定模板的保存位置并在“文件名”框中键入新建模板的名称，单击“保存”按钮即可。

✍ 这里所说的已有模板包括“Normal”模板。

2. 根据已有文档新建模板

如果要新建的模板与某个已有文档很相似，则可在该文档的基础上来新建模板。

首先打开与要新建的模板相似的文档，然后在文档的编辑界面下按照要创建的模板的要求对文档的内容和设置进行修改。修改完成后，打开菜单栏中的“文件”菜单，并选择其中的“另存为”命令，将出现图 2-15 所示的“另存为”对话框。与前面不同的是此时“保存类型”框是可选的，并且初始选中为“Word 文档”。单击“保存类型”框右端的下拉箭头并从下拉列表中选中“文档模板”，然后在“文件名”框中键入新模板的名称并在“保存位置”框中指定新建模板的保存位置，单击“保存”按钮即可。

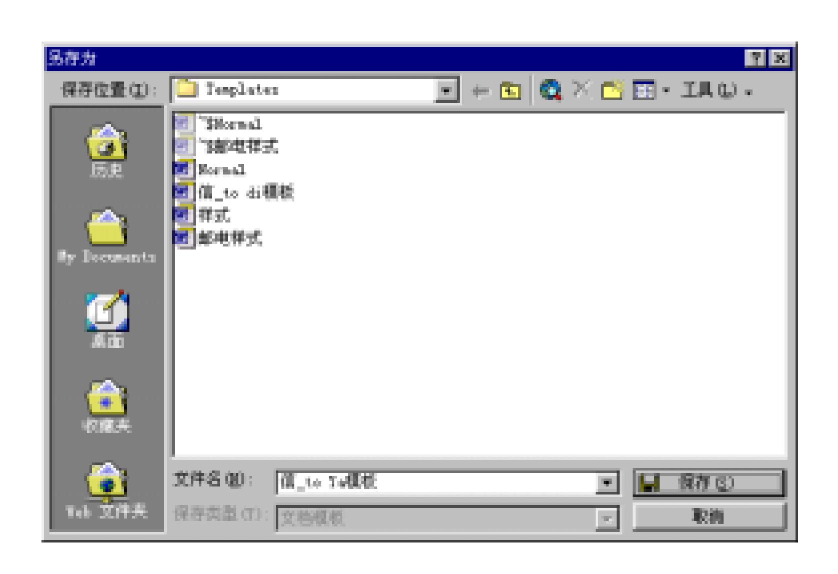


图 2-15 “另存为”对话框

3. 新建模板的保存位置

不管用哪种方法新建了模板，在“另存为”对话框中都需要指定新模板的保存位置。

通常将新建模板保存在“Templates”文件夹中（路径通常为“C:\Windows_文件夹\Application Data\Microsoft\Templates”或“C:\Windows_文件夹\Profiles\用户名\Application Data\Microsoft\Templates”)。这样该模板将出现在“新建”对话框的“常用”选项卡中。如果用户在“Templates”文件夹中建立了子文件夹并将新建模板保存到此子文件夹中，则在“新建”对话框中将出现一个以子文件夹名称命名的选项卡，新建模板将出现在这个选项卡中。如图 2-16 所示。

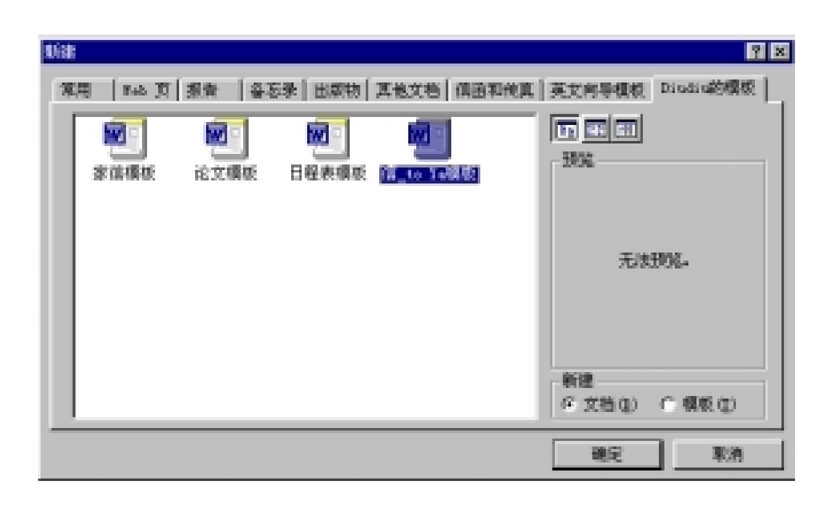


图 2-16 “新建”对话框中用户自定义的选项卡

如果新建模板不保存在“Templates”文件夹或其子文件夹中，它将不会出现在“新建”对话框中。

✍ 保存在“Templates”文件夹中的任何文档文件（扩展名为.DOC）都可以作为模板使用。

2.4　分隔符

Word 2000 提供了多种分隔符，如分页符、分栏符、分节符、换行符等。利用它们可以使文档版面格式更加丰富，文档层次更加分明。同时，分隔符的使用，将给文档的编辑及阅读带来很大的方便。

打开菜单栏中的“插入”菜单，并选择其中的“分隔符”命令，将出现如图 2-17 所示的“分隔符”对话框，利用这个对话框可以在文档中插入分页符、分栏符、换行符以及分节符等分隔符，从而完成对文档的分页、分栏、换行以及分节操作。



图 2-17　“分隔符”对话框

本节将主要针对分页符、分节符以及分栏符进行介绍。

2.4.1 分页符

分页符是文档的一页与其他页隔开的标志，被设置在上一页结束与下一页开始之间，分为自动分页符与人工分页符两种。

当页面中文字或者图形填满一页后，Word 2000 将会插入一个自动分页符（又叫软分页符），并开始新的一页。如果用户想在尚不满一页的某个指定位置处强制分页，则必须插入人工分页符（又叫硬分页符）。

插入人工分页符时，首先将光标置于要强制分页的位置，然后按照前面的方法打开图 2-17 所示的“分隔符”对话框，选中“分页符”选项后单击“确定”按钮即可。也可按下快捷键“Ctrl+Enter”在光标处插入分页符。

在页面视图、打印预览视图状态下，分页符后的文本将出现在新页中。在普通视图中，自动分页符显示为横穿页面的单点划线，而人工分页符则显示为标有“分页符”字样的单点划线。

2.4.2 分节符

为了便于文档的格式化以及阅读的方便，可以利用分节符将文档分成多个“节”。

Word 2000 中各个“节”的起始位置和终止位置是由用户自己定义的。节既可以是一个段落，也可以只是段落中的某一部分，还可以是多个段落，甚至可以是整篇文档（不进行分节时 Word 2000 默认整个

文档为一节)。总之，节的设置是非常灵活的，将文档分为节的功能给文档的制作带来了极大的方便。

分节符是节的结束标记。在分节符中保存了节的各种格式设置元素。文档分节后，可以对各节独立地进行格式化而不会影响其他节的格式。

插入分节符时，首先将光标置于要分节的位置，然后按照前面的方法打开图 2-17 所示的“分隔符”对话框，并且在“分节符类型”栏中选中适当的分节符类型后，单击“确定”按钮即可。

“分节符类型”栏共包括四个选项：“下一页”、“连续”、“偶数页”以及“奇数页”。选中“下一页”，将插入一个分节符，新节从下一页开始；选中“连续”，将插入一个分节符，新节从本页开始；选中“偶数页”或“奇数页”，将插入一个分节符，新节从下一个偶数页或奇数页开始。

分节符显示为包含有“分节符”字样及分节符类型说明的双虚线。

2.4.3 分栏符

在介绍分栏符之前，首先对分栏排版进行简单的介绍。

利用 Word 2000 提供的分栏功能可以为文档版面设计出报纸、杂志中常见的排版效果。

要创建分栏，在切换到页面视图状态后，首先应当选定要设置分

栏格式的文本。例如，选定整个文档或其中的某几个段落。如果要将已有的节设置为分栏格式，则可单击一节或选择多节。打开菜单栏中的“格式”菜单，并选择其中的“分栏”命令，将出现如图 2-18 所示的“分栏”对话框。在此可以进行分栏的栏数、各栏的栏宽、相邻栏间的间距以及是否添加分隔线等设置。



图 2-18　“分栏”对话框

✍ 文档分栏后，如果要要调整栏宽和栏间距，除了再次打开“分栏”对话框进行重新设置外，也可以直接拖动水平标尺上的分栏标记进行调整。后一种方法可以更加直观地看到调整后的效果。

文档分栏之后，文字或者图形总是先填满一栏后再自动从下一栏的顶端开始。如果用户想在尚不满一栏的某个指定位置处强制开始新栏，则须插入分栏符。

插入分栏符时，首先将光标置于要强制分栏的位置，然后按照前面的方法打开图 2-17 所示的“分隔符”对话框，选中“分栏符”选项后单击“确定”按钮即可。

分栏符显示为包含有“分栏符”字样的一条单点划线。

- 如果看不到人工分页符、分节符、分栏符等，可单击“常用”工具栏中的“显示/隐藏编辑标记”按钮使之显示出来。
- 选中分页符、分节符及分栏符标志后，按下“Del”键，可以将其删除。

2.5 文档注释

为了便于阅读者更好地理解文档，给文档加上必要的注释是一种很好的作法。另一方面，用户在阅读文档时，可能希望在某些地方写下自己的心得体会。Word 2000 允许对文档进行注释，以下将对添加批注以及脚注和尾注进行介绍。

2.5.1 批注

批注是文档的作者或审阅者给文档添加的注释或注解。在文本区以及页眉和页脚区域均可插入批注。由于加入的注释都是隐含的文本，因此不会改变原有文档。

1. 插入批注

插入批注时，首先将光标移到要插入批注的位置或者选定要加批注的文字，然后打开菜单栏中的“插入”菜单，并选择其中的“批注”命令，Word 2000 将用黄色作为批注文本的底纹并在光标处插入批注引用标记，同时在屏幕底端显示“批注”窗格，如图 2-19 所示。

批注标记由批注者的姓名缩写和批注的编号组成（例如图中的[sy1]），在文档中以隐藏文字的格式插入，打开批注窗格时将显示批注标记。如果关闭批注窗格后看不到批注标记，则可单击“常用”工具栏中的“显示/隐藏编辑标记”按钮来显示批注标记。Word 2000 使用不同的颜色来标识不同审阅者的批注引用标记。

批注窗格位于屏幕底端，在其中可以键入、修改批注文字。批注文字的编辑方法与普通文本相同。单击“批注来源”框右端的下拉箭头可以从下拉列表中查看批注者。如果计算机带有声卡和麦克风，单击“插入声音批注”按钮就可以录制并插入声音批注。完成批注后，单击“关闭”按钮可以关闭批注窗格。如果想再次打开批注窗格，可以打开菜单栏中的“视图”菜单，并选择其中的“批注”命令。也可双击任一批注标记打开批注窗格。

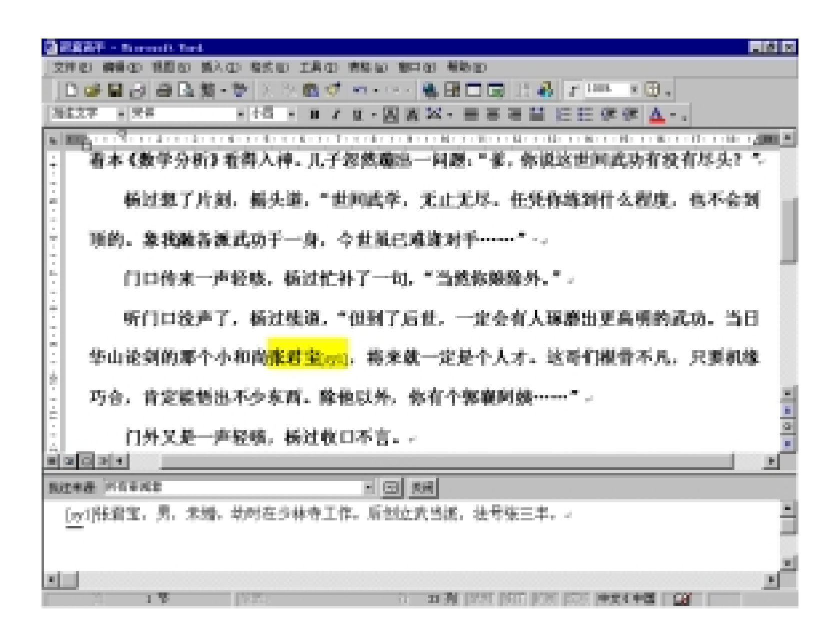


图 2-19 插入批注

2. 查看批注

在 Word 2000 中可以联机查看批注。将鼠标指针在有黄色底纹的文字上停留片刻，文字上方将出现类似于图 2-20 所示的屏幕提示，其中显示了批注内容及批注者姓名。

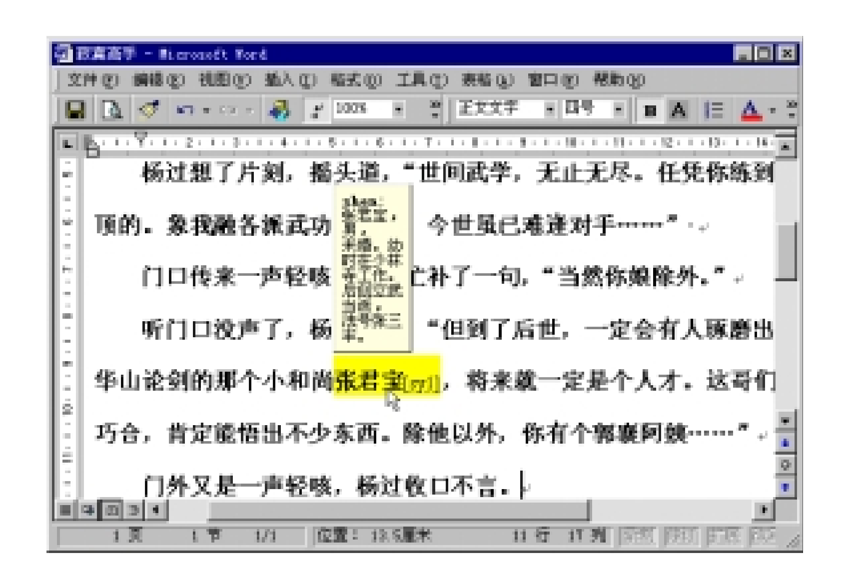


图 2-20　联机查看批注

✍ 如果不能联机查看批注，可单击“工具”菜单中的“选项”命令，并从“视图”选项卡中选中“屏幕提示”复选框。

也可在批注窗格中查看批注。在批注窗格中查看批注时可以审阅文档中所有的批注，同时还可以对批注进行编辑。

3. 删除和打印批注

删除批注时，在文档中选定要删除的批注标记后，按下“Del”键即可。

如果想要在打印文档的同时把批注也打印出来，可以打开菜单栏中的“工具”菜单，并选择其中的“选项”命令打开“选项”对话框，再从“打印”选项卡的“打印文档的附加信息”栏中选定“批注”复选框。

2.5.2 脚注和尾注

脚注和尾注主要用于在打印文档中为文档中的文本提供解释、批注以及相关的参考资料。其中脚注出现在文档中每一页的底端，用来对本页中的文档内容进行注释说明;而尾注一般位于整个文档的结尾，常用来注明文档中引用过的文献。

1. 插入脚注和尾注

插入脚注或尾注时，首先将光标移到要插入脚注或尾注的位置，然后打开菜单栏中的“插入”菜单，并选择其中的“脚注和尾注”命

令，此时将出现如图 2-21 所示的“脚注和尾注”对话框，在此可以对脚注和尾注进行设置。



图 2-21　“脚注和尾注”对话框

首先在“插入”栏中选中“脚注”或“尾注”，然后在“编号方式”栏中指定编号方式。若选中“自动编号”，Word 2000 将自动给所有的脚注和尾注连续编号；若选中“自定义标记”，须在框中键入要使用的标记（不超过 10 个字符）。如果想使用特殊符号，可单击“符号”按钮打开“符号”对话框并从中选择所需符号。

如果用户希望对脚注或尾注的位置和编号格式进行设置，可单击“选项”按钮，此时将出现如图 2-22 所示的“注释选项”对话框，其中包括“所有脚注”和“所有尾注”两个非常相似的选项卡，分别用于对脚注和尾注进行设置。

在“所在位置”框的下拉列表中可以指定脚注或尾注的位置。对于脚注选项卡，有“页面底端”和“文字下方”两个选项；对于尾注

选项卡，则有“文档结尾”和“节的结尾”两个选项。在编号格式的下拉列表中可以指定编号的字体、字号以及编号方式，例如“1，2，3…”、“A，B，C…”、“a、b、c…”、“ⅰ、ⅱ、ⅲ…”、“甲、乙、丙”等。在“起始编号”框中可以选定或者键入第一个脚注或尾注的序号，Word 2000 的默认值为 1。在“编号方式”栏中可以选定适当的编号方式，对于脚注选项卡，有“连续编号”、“每节重新编号”及“每页重新编号”三个选项；对于尾注选项卡，则有“连续编号”及“每节重新编号”两个选项。完成设置后，单击“确定”按钮返回“脚注和尾注”对话框。



图 2-22　“注释选项”对话框

✍ 在“注释选项”对话框中进行的编号格式及起始编号的设置，只有当在“脚注和尾注”对话框中选中“自动编号”后才有效。

单击“脚注和尾注”对话框的“确定”按钮后，Word 2000 将在光标处插入注释编号，并在文档中添加脚注或尾注。如果文档在页面视图状态下显示，注释文本将显示在要打印的位置处，此时可以同编辑普通文本一样对脚注和尾注进行编辑，包括脚注和尾注的键入、修改、格式设置等。如果文档在普通视图状态下显示，注释文本将显示屏幕下方相应的注释窗口中，在窗口中可以对脚注或尾注进行编辑。单击脚注或尾注窗口的“关闭”按钮可以关闭脚注或尾注窗口。如果想再次打开窗口，可以打开菜单栏中的“视图”菜单，并选择其中的“脚注”命令。也可双击任一注释引用标记打开脚注或尾注窗口。

2. 查看脚注和尾注

在 Word 2000 中可以联机查看脚注和尾注。与联机查看批注类似，将鼠标指针在注释引用标记上停留片刻，文字上方将出现屏幕提示，其中显示了脚注或尾注的注释内容。

在页面视图状态下，还可以在与打印文档相同的位置直接查看到脚注和尾注。在普通视图状态下也可在脚注或尾注窗口中查看注释文本。在脚注窗口的“脚注”框下拉列表中选择“所有尾注”可以切换到尾注窗口，同样可从尾注窗口切换到脚注窗口。

3. 删除脚注和尾注

删除脚注或尾注时，在文档中选定要删除的脚注或尾注的注释引用标记后，按下“Del”键即可。

如果想要在打印文档的同时把批注也打印出来，可以打开菜单栏中的“工具”菜单，并选择其中的“选项”命令打开“选项”对话框，再从“打印”选项卡的“打印文档的附加信息”栏中选定“批注”复选框。

4. 脚注和尾注间的转换

有时用户希望将已经插入的脚注转换为尾注，或将尾注转换为脚注。在 Word 2000 中既可以只对单个注释进行转换，也可转换所有的注释。

如果想把所有的注释都转换成脚注或尾注，首先按照前面的方法打开图 2-22 所示的“注释选项”对话框，单击其中的“转换”按钮，将出现如图 2-23 所示的“转换注释”对话框。其中包括“脚注全部转换成尾注”、“尾注全部转换成脚注”以及“脚注和尾注相互转换”三个选项，选中所需选项后单击“确定”按钮即可。

将某一个注释转换成脚注或尾注时，用鼠标右键单击要转换的注释，将弹出类似于图 2-24 所示的快捷菜单，选择其中的“转换至脚注”或“转换至尾注”命令即可。

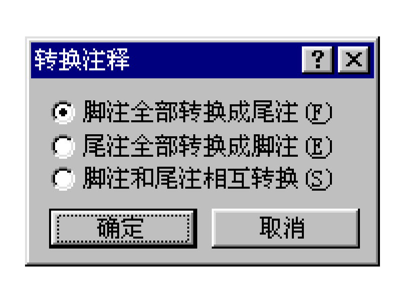




图 2-23　“转换注释”对话框　　　　图 2-24　“转换注释”快捷菜单

2.6　表格

利用表格可以组织和显示需要分门别类的信息。与普通文本相比，表格将使文档内容更加简洁，条理更加清楚。如果可以将某些冗长的文本用精美的表格来代替，必定能给文档的版面外观带来很好的修饰效果。Word 2000 提供了强大的表格功能，用户不仅能够利用这些功能非常方便地制作出各种形式的表格，还可以完成诸如排序、计算等工作。本节将对表格的格式化及表格的一些自动功能进行介绍。

2.6.1　格式化表格

为了使表格的结构更加合理，外观更加精美，常常需要对表格进行格式化。

格式化表格包括格式化表格本身和格式化表格内容两部分。对单

元格中的文本内容进行格式化时，方法与格式化普通文本相同，因此以下重点介绍的将是对表格本身的格式化。

1. 调整表格尺寸

调整表格尺寸时，将指针停留在表格上，此时表格右下角将出现一个小方框，这就是表格的尺寸控点。将指针停留在表格尺寸控点上会指针变为一个双向箭头，此时按住鼠标左键可将表格的边框拖动到所需尺寸大小。

按以上方法改变表格尺寸后，表格各单元格的尺寸将按照相同的比例被缩放。用户也可以只对表格中局部的行、列尺寸进行调整。方法是将指针停留在要更改其宽度的行、列的边框上，直到指针变为“←||→”状或“←||→”状后，按住鼠标左键拖动边框，直到得到所需的行高或列宽为止。

✍ 也可拖动垂直标尺或水平标尺上的行标记或列标记来调整行高或列宽。

以上介绍的方法可以方便快捷地调整表格尺寸，缺点是可能不够精确。为了能准确地改变表格尺寸，可以在“表格属性”对话框里进行设置。打开菜单栏中的“表格”菜单，并选择其中的“表格属性”命令，将出现如图 2-25 所示的“表格属性”对话框。单击“行”或“列”标签可以分别切换到“行”、“列”选项卡进行设置。

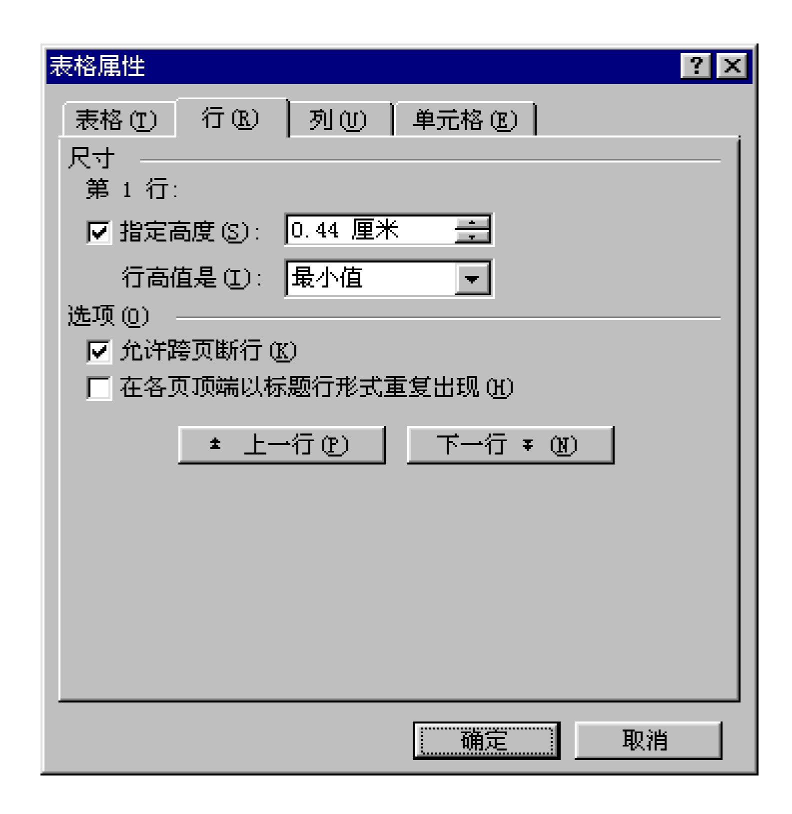


图 2-25　“表格属性”对话框中的“行”选项卡

以“行”选项卡为例，在“指定高度”框中可以键入或选定行高度。单击“上一行”或“下一行”按钮可以选择不同的行进行尺寸设置。设置完成后单击“确定”按钮即可。在“列”选项卡中的设置与此类似。

✍ 要使表格中的列根据表格内容自动调整宽度，首先将光标移到要进行此项设置的列中，然后打开菜单栏中的“表格”菜单，并选择“自动调整”子菜单中的“根据内容调整表格”命令即可。

2. 插入行、列或单元格

实际应用过程中，往往需要对最初建立的空表格结构进行一定程度的调整，例如需要插入或者删除某些行、列或单元格等。在 Word 2000 中建立空表格之后，可以随时对表格的结构进行修改。

向表格中插入行或列时，首先选定与要插入的行或列数目相同的行或列，然后打开菜单栏中的“表格”菜单，并选择其中的“插入”命令，将出现如图 2-26 所示的二级菜单。从中可以选择在选定行的上方或下方、或者在选定列的左侧或右侧插入与选定行、列数目相同的行、列。



图 2-26 “插入”二级菜单

- 如果想在表格中插入一行，也可以将光标移到需要插入行的上一行末尾的结束标记处并按回车键。
- 如果想在表格的末尾插入一行，还可以将光标移到表格最后一个单元格中结束标记处并按“Tab”键。

向表格中插入单元格时，首先选定与要插入的单元格数目相同的单元格，然后在图 2-26 所示的“插入”二级菜单中选择“单元格”命

令，此时将出现如图 2-27 所示的对话框。选定“左侧单元格右移”，表示新单元格插入在选定单元格的左方；选定“活动单元格下移”，表示新单元格插入在选定单元格的上方。



图 2-27 “插入单元格”对话框

- 若在“插入单元格”对话框中选择“整行插入”或“整列插入”，则分别将在选定单元格所在行的上方或所在列的右侧插入与选定单元格的行、列数目相同的行或列。

3. 删除行、列或单元格

删除表格中的行、列或单元格与插入行、列或单元格一样可以改变表格结构。删除行、列或单元格时，不仅删除了表格中的内容，同时行、列或单元格本身也被删除。

删除行、列时，首先选定要删除的行、列，然后打开菜单栏中的“表格”菜单，并选择“删除”子菜单中的“行”或“列”命令。

- 删除行或列时，也可在选定行或列后按下“Ctrl+X”键或者单击“编辑”菜单中的“剪切”按钮。

删除单元格时，首先选定要删除的单元格，然后打开菜单栏中的“表格”菜单，并选择“删除”子菜单中的“单元格”命令，此时将

出现如图 2-28 所示的“删除单元格”对话框。从中选定“右侧单元格左移”或“下方单元格上移”选项可以指定删除单元格后的表格格式。



图 2-28 “删除单元格”对话框

✍ 若在“删除单元格”对话框中选择“整行删除”或“整列删除”选项，则分别删除选定单元格所在的行或列。

4. 合并与拆分

对表格可以进行合并与拆分。既可以合并或拆分整张表格，也可只对表格的一部分进行合并或拆分。

合并表格单元格时，被合并的单元格应当是相邻的（包括相邻行或者相邻列）。首先选定要合并的单元格，然后打开菜单栏中的“表格”菜单，并选择其中的“合并单元格”命令即可。合并过程及合并后的表格结构如图 2-29 所示。

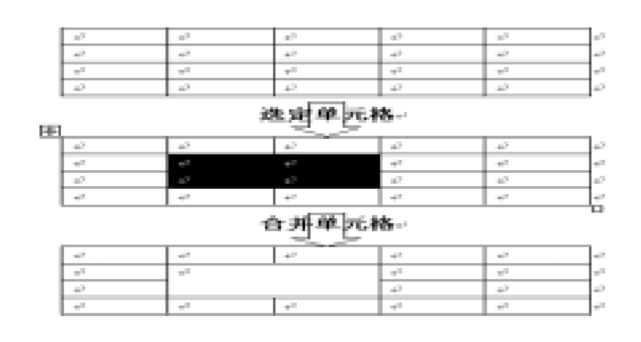


图 2-29　合并单元格

如果要将两张表格合并在一起，只需将两张表格之间的所有文字、图片、空格等通通删除即可。

拆分单元格时，首先选定要拆分的单元格（同合并单元格类似，这些单元格也应当是相邻的）。然后打开菜单栏中的“表格”菜单，并选择其中的“拆分单元格”命令，此时将出现如图 2-30 所示的“拆分单元格”对话框。从中指定要将单元格拆分成的行、列数后，单击“确定”按钮即可。



图 2-30　“拆分”单元格对话框

✍ 如果选定的是多个单元格，则复选框“拆分前合并单元格”变为可选。若清除复选框，则指定的拆分成的行、列数将应用于每一个选定的单元格。若选中该复选框，Word 2000 将先把所选定的多个单元格合并成一个单元格，然后按照指定的行、列数进行拆分，这可用于快速重设表格。

将一个表格拆分成两个表格时，首先将光标移到要拆分成的第二个表格的首行任意单元格中，然后打开菜单栏中的“表格”菜单，并

选择其中的“拆分表格”命令即可。也可按快捷键“Crtl+Shift+Enter”拆分表格。

5. 表格与文本的转换

在 Word 2000 中可以很方便地实现表格与文本之间的相互转换。

将表格转换成文字时，首先选定表格或将光标移到表格中任意位置，然后打开菜单栏中的“表格”菜单，并选择“转换”子菜单中的“表格转换成文字”命令，将出现如图 2-31 所示的“将表格转换成文字”对话框。从中可以指定用来分隔文字位置的分隔符，如段落标记、制表符、逗号，也可选中“其他字符”选项后在框内键入其他分隔符。指定分隔符后单击“确定”按钮即可。



图 2-31　“将表格转换成文字”对话框

表格转换成文字后，Word 2000 将用段落标记代替行尾标记，用指定的分隔符来替换列的边界。转换前的表格和转换后的文字格式如图 2-32 所示。

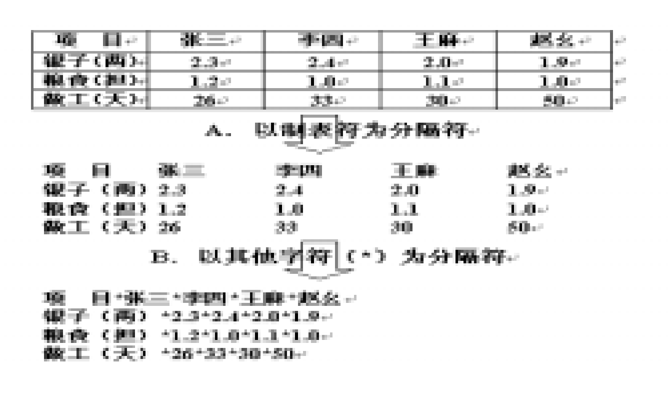


图 2-32　将表格转换成文字

Word 2000 将文字转换成表格的功能也在很多场合下带来了极大的方便。将文字转换成表格时，首先必须对文字进行格式化。也就是说，对要转换成表格的文本，在转换之前使用段落标记标明新的一行，并且使用表格分隔符标记新的列开始的位置（即在要划分列的位置插入所需的分隔符）。

完成对文字的格式化之后，选定要转换的文字，打开菜单栏中的“表格”菜单，并选择“转换”子菜单中的“文字转换成表格”命令，将出现如图 2-33 所示的“将文字转换成表格”对话框。“表格尺寸”栏中给出了转换后的表格的行列数，一般来说，只要对文字的分隔符设置是正确的，行、列数也就会与预想相同。在“‘自动调整’操作”栏中若选择“固定列宽”，则可在框中选定或键入列的宽度，默认值“自动”表示以左右页边距距离为表格总的宽度，并且各列宽度相同；也可选中“根据窗口调整表格”或“根据内容调整表格”使列宽能自动调整。在文档“文字分隔位置”栏中，根据格式化文字时所使用的分

隔符选中相应选项或选中“其他字符”后在框内键入所使用的分隔符。

图 2-33 “将文字转换成表格”对话框

文字转换成表格后，Word 2000 会将段落标记所在的位置作为行的起点，将指定的分隔符所在的位置作为列的起点。

6. 设置表格位置与文字环绕方式

考虑到文档版面的美观，通常需要对表格的位置进行调整。例如，很多情况下表格被放置在页面的中央而不是左对齐页边距。另外，有时表格与文字的位置关系也需要调整。这些都可以在“表格属性”对话框里进行设置。

打开菜单栏中的“表格”菜单，并选择其中的“表格属性”命令，将出现如图 2-34 所示的“表格属性”对话框（其中的“行”、“列”选项卡已在前面进行了介绍），单击“表格”标签切换到“表格”选项卡

进行设置。

在“对齐方式”栏中包括了“左对齐”、“居中”和“右对齐”三个选项，分别设置的表格位置为：表格的左边界对齐左页边距、表格放置在左右页边距的中央以及表格的右边界对齐右页边距。如果用户想把表格放置在上述位置以外的区域，则可先选中“左对齐”方式，然后在“左缩进”框中选定或键入表格左边界与左页边距之间的距离（以厘米为单位）。

在“文字环绕”栏中可以设置文字是否环绕表格。如果选中了“环绕”，则可单击“定位”按钮并在弹出的“表格定位”对话框中对表格与环绕文字之间的距离、表格是否随文字移动等属性进行设置。



图 2-34　“表格属性”对话框中的“表格”选项卡

✍ 单击“表格”选项卡中的“边框和底纹”按钮将弹出“边框和底纹”对话框，从中可以对表格的边框和底纹进行设置。由于本章稍后将专门对设置边框和底纹进行讨论，故此处从略。

7. 设置单元格对齐方式

在默认情况下，Word 2000 将文字与单元格的左上角对齐。用户也可以更改单元格中文字的对齐方式。

更改单元格中文字的对齐方式时，首先选定要设置文本对齐方式的单元格（单个或多个），然后单击“表格和边框”工具栏中的“单元格对齐方式”按钮右端的下拉箭头，将出现一个“单元格定位”下拉菜单，其中共提供了九种对齐方式，单击所需方式即可。

2.6.2　表格计算

Word 2000 的表格提供了计算功能，可以完成加、减、乘、除、求最大值或最小值等常用运算。这给表格数据的处理带来了极大的方便。

计算中必然要引用数据。Word 2000 并不是直接引用各个单元格中的数据，而是根据各个单元格的参数（编号）来引用单元格数据的。Word 2000 是按照单元格在表格中的位置来为单元格分配参数的，参数由一个英语字母和一个阿拉伯数字组成，其中字母代表列、数字代

表行，如图 2-35 所示。在公式中引用单元格时，用逗号分隔单个单元格，而选定区域的首尾单元格之间用冒号分隔，例如要计算图 2-35 中 b1、c1、b2、c2 四个单元格的和，则公式应为“SUM(b1，c1，b2，c2)”或者“SUM(b1:c2)”。

a1	b1	c1	d1
a2	b2	c2	d2
a3	b3	c3	d3

图 2-35　单元格的参数分布

✍ 可以使用只有字母或数字的区域分别表示引用整列或整行，例如“a:a”表示表格的第一列，“1:1”表示表格的第一行，又如公式“MAX(1:1，2:2)”或“MAX(a1:d2)”均表示求图中前两行中的最大值。

在表格中进行计算时，首先将光标移到要放置计算结果的单元格，然后打开菜单栏中的“表格”菜单，并选择其中的“公式”命令，此时将出现如图 2-36 所示的“公式”对话框。在“公式”框内键入或选定所用的计算公式，并在“数字格式”框内选定或键入计算结果所用的数字格式，单击“确定”按钮后，Word 2000 将在光标处插入以指定的数字格式表示的计算结果。

公式

公式(F):

=SUM(LEFT)

数字格式(N):

粘贴函数(U):　粘贴书签(B):

确定　取消

图 2-36　“公式”对话框

实际计算中往往并不是直接在“公式”框内键入计算公式，利用 Word 2000 提供的函数可以更方便地给出计算公式。例如键入“=(a1+a2+a3)/3”可以简化粘贴求平均值函数“AVERAGE”再键入“a1，a2，a3”或“a1:a3”。粘贴函数时单击“粘贴函数”框右端的下拉箭头，可以看到其下拉列表中包括了多种常用函数，例如求和函数“SUM”、求绝对值函数“ABS”、取整函数“INT”、求或函数“OR”等等。选定某个函数后它将出现在“公式”框中。利用这些函数可以组成各种计算关系。

下面举一个简单的例子来说明如何在表格中进行计算。

现有甲、乙、丙三人的成绩表如图 2-37 所示。其中已有的数据为三人的各科成绩，现在要在最后一行中计算出各自的总分，并在最后一列中统计出各科目三人的平均分以及三人总分的平均分（最后一个单元格）。

甲、乙、丙成绩表

语文	80	82	85	
数学	92	94	90	
外语	87	85	90	
总分				

图 2-37　原始成绩表

首先计算甲的总分，将光标移到放置计算结果的单元格即第四行第二列中，然后调出图 2-36 所示的“公式”对话框。可以看见在“公式”框中建议采用公式“=SUM(ABOVE)”进行计算（如果光标所在单元格位于某行数值的最右端，Word 将建议采用公式“=SUM(LEFT)”进行计算）。用户也可从“粘贴函数”框里选择求和函数“SUM”然后在“公式”框里函数的括号中键入要进行求和运算的单元格参数将公式补充完整，当然也可以直接在“公式”框里键入公式。本例中与“=SUM(ABOVE)”等价的公式形式还有“=SUM(b1，b2，b3)”、“=SUM(b1:b3)”以及“= b1+b2+b3”等。最后从“数字格式”框的下拉列表中选择计算结果的数字格式，本例中总分为整数，故选择“0”。确认计算公式正确后单击“确定”按钮，则光标所在的单元格中将出现总分计算结果“259”。按照同样的方法计算出乙和丙的总分。

接下来在最后一列中统计各科目或总分的三人平均分。首先计算语文平均分，将光标移到放置计算结果的单元格即第一行第五列中，然后在“公式”对话框里设置计算公式。本例中等价的一些计算公式

的形式有“=AVERAGE(b1，c1，d1)”、“=AVERAGE(b1:d1)”、“=(b1+c1+d1)/3”、“=SUM(b1，c1，d1)/3”以及“= SUM(b1:d1)/3”等。从“数字格式”框的下拉列表中选择计算结果的数字格式时，本例中平均分保留两位小数，故选择“0.00”。单击“确定”按钮后，光标所在的单元格中将出现平均分计算结果“82.33”。按照同样的方法计算出数学、外语以及总分的平均分。最终的计算结果如图 2-38 所示。

甲、乙、丙成绩表

语文	80	82	85	82.33
数学	92	94	90	92.00
外语	87	85	90	87.33
总分	259	261	265	261.67

图 2-38　计算结果

2.6.3　表格排序

Word 2000 可以自动对表格进行排序，从而为用户省却了大量繁冗的工作。例如，在成绩表中按照平均分的高低进行排序，在通讯录中按照姓名的拼音进行排序等。

对表格排序时可以有多种排序规则，常用的有按笔画排序、按数字排序、按日期排序以及按拼音排序。需要注意的是，按数字排序时 Word 2000 忽略数字以外的其他所有字符；按日期排序时，Word 2000 将连字符、斜杠（/）、逗号、句点和冒号等符号默认为日期分隔符；

按拼音排序时，Word 2000 将以标点或符号（如!、#、$、%、&等）开头的条目排在最前面，接下来是以数字开头的条目，然后是以字母开头的条目，最后才将以汉字开头的条目按拼音排序。

- ✍ 按拼音排序时，如果 Word 2000 如果无法识别某个日期或时间，则升序时把该项排在开头，降序时把该项排在结尾。
- ✍ 按照某种排序规则进行排序时，如果两个条目的首字符相同，Word 2000 将比较各条目中的后续字符以决定排列次序。

对于每一种排序规则，都有升序和降序两种顺序相反的排列方法。例如按拼音排序时，从 A 到 Z 为升序，从 Z 到 A 为降序；按日期排序时，日期由早到晚为升序，日期由晚到早为降序。

对整张表格进行排序时，首先选定表格或者将光标移到表格中任意位置，然后打开菜单栏中的“表格”菜单，并选择其中的“排序”命令，此时将出现如图 2-39 所示的“排序”对话框。单击“排序依据”框右端的下拉箭头并从其下拉列表中选择要用作表格行排序基准的列，在“类型”框的下拉列表中指定排序规则（按笔画、数字、日期或是拼音排序），并选中“递增”或“递减”选项指定按升序或是按降序排序。如果需要将多列指定为排序基准，可在“然后依据”栏中进行设置。在“列表”栏中选定“无标题行”选项可对表格中所有行排序，选定“有标题行”选项则可对除首行外的整个表格排序。完成排

序设置后，单击“确定”按钮即可。



图 2-39　“排序”对话框

对表格中单独的一列或列中的多个单元格进行排序时，首先选定要进行排序的列或单元格，然后单击“排序”对话框中的“选项”按钮，此时将出现如图 2-40 所示的“排序选项”对话框，在“排序选项”栏中选定“仅对列排序”复选框后，单击“确定”按钮返回“排序”对话框。

✍ 在“排序选项”对话框中还可对分隔符形式、排序语言以及是否区分大小写等选项进行设置。



图 2-40　“排序选项”对话框

✍ 选定整张表格或选定表格的行、列、单元格后单击鼠标右键可以调出如图 2-41 所示的一些快捷菜单。利用快捷菜单可以非常方便地访问本节中介绍过的很多命令。

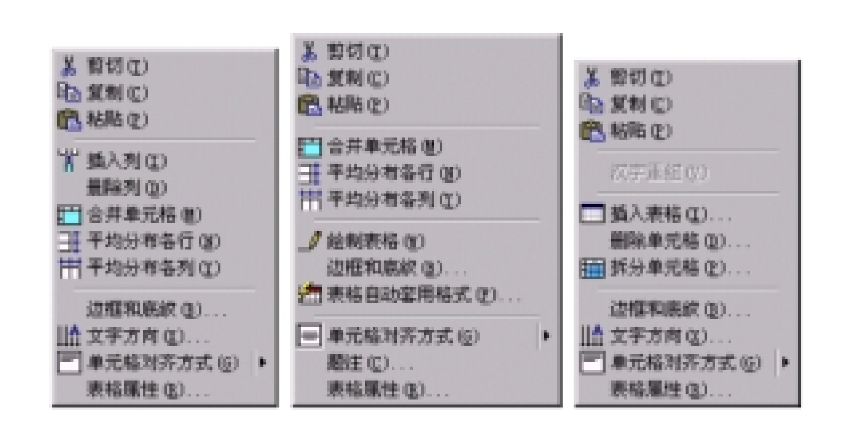


图 2-41　快捷菜单

2.7　图片和图形

图片和图形在文档的修饰中所起的作用是不言而喻的。作为一种字处理软件的 Word 2000，其功能早已远远超出了单纯的字处理的范

畴，它支持在文档中插入图片和图形并能对图片和图形进行处理，同时 Word 2000 还提供了绘图功能。这使得用户可以利用 Word 2000 制作出各种生动形象的文档。

2.7.1 图片和图形的区别

Word 2000 中的图形对象包括自选图形、曲线、直线以及艺术字图形对象等。这些对象都是 Word 2000 文档的一部分，因此可以使用“绘图”工具栏对图形进行编辑，以更改或改善其外观效果。

而图片则是由其他文件创建的图形，包括剪贴画、位图、扫描的照片或图片等。使用“图片”工具栏的工具和“绘图”工具栏的部分工具可以更改和增强图片。但是在某些情况下，必须取消图片的组合并将其转换为图形对象后才能使用“绘图”工具栏的工具进行编辑（很多图片的组合是无法取消的）。

2.7.2 插入图片

在文档的编辑过程中，可以随时插入已有的图片。这些图片可以是 Word 2000 自带的图片，也可以是用户利用其他软件制作出的或者从其他任何位置下载的图片。

打开菜单栏中的“插入”菜单，并选择其中的“图片”命令，将出现如图 2-42 所示的二级菜单，利用这个菜单可以在光标所在处插入图片（包括剪贴画和其他路径下的图片）。



图 2-42　“图片”二级菜单

2.7.3　绘制自选图形

有的时候，现有的图片并不能满足文档的要求，此时用户需要自己制作图形。可以利用专门的图形制作软件来绘制图形，再将其插入到文档中，也可以利用 Word 2000 提供的图形绘制功能直接在文档中绘制所需的图形。在很多情况下 Word 2000 的绘制自选图形功能完全能够满足需要。

Word 2000 附带了一组现成的可在文档中使用的自选图形，其中既包括了矩形、圆形这样的基本图形，也包括各种各样的线条和连接符、箭头、流程图符号、星形、旗帜形和标注等。利用绘制自选图形的功能可以非常方便地绘制出常用的图形，并且可以通过对自选图形的大小调整、旋转、翻转、设置颜色和组合制造出各种复杂形状的图形。

绘制自选图形时，打开菜单栏中的“插入”菜单，并选择图 2-42 所示的“图片”二级菜单中的“自选图形”命令，此时将出现一个“自选图形”工具栏，其中包括了线条、基本形状、箭头、流程图元素、

星与旗帜以及标注等几种形状类型按钮。单击每类形状的按钮将出现下拉列表，其中包括了一些基本形状，如图 2-43 所示。

单击适当的自选图形形状后，鼠标指针将变为“十”状，将此指针移到要绘制图形的位置后，单击鼠标左键，Word 2000 将按照预定义大小绘制出所选的自选图形；也可以按住鼠标左键同时拖动鼠标将图形调整至所需大小。

✍ 拖动图形的同时按住“Shift”键可以保证自选图形长宽的比例不变。

✍ 绘制图形必须在页面视图状态下进行。

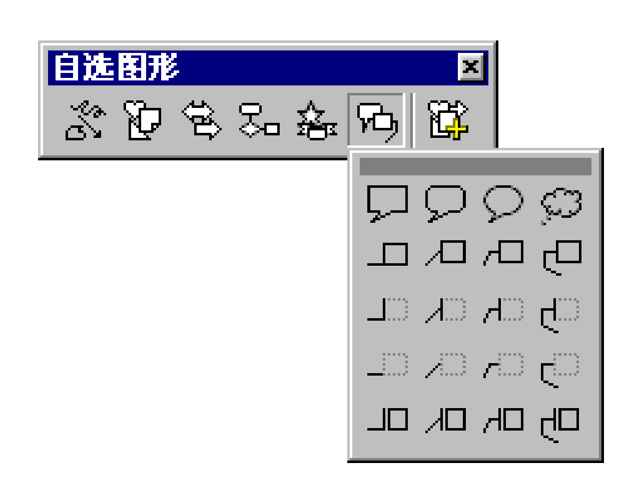


图 2-43　“自选图形”工具栏及其下拉列表

利用 Word 2000 的绘制自选图形功能绘制出的一些图形如图 2-44 所示。

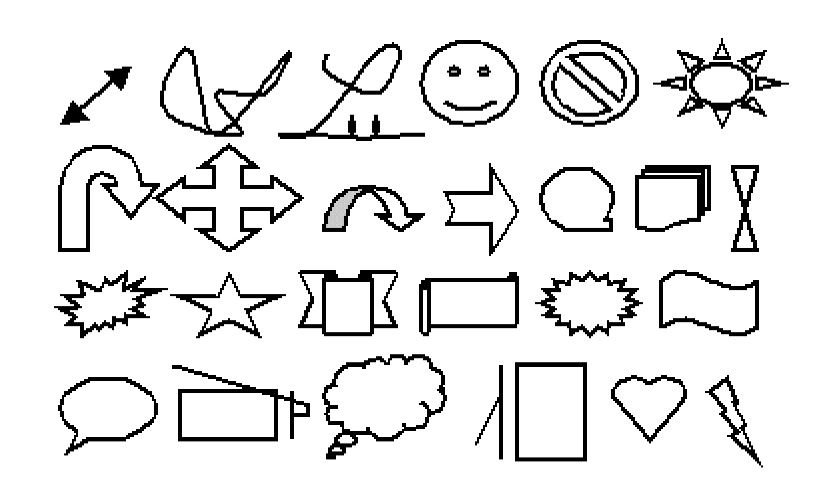


图 2-44　利用自选图形功能绘制的图形

2.7.4　插入艺术字

对于一篇外观精美的文档，除了要进行高质量的格式编排、恰如其分地插入表格或图片外，对文字的美化也是必不可少的一个环节。尤其是在海报、广告或者出版物封面等文档的制作中，只使用普通的文字形式是收不到应有效果的;而艺术字具有更强烈的文字视觉效果，可以更有效地修饰文档。

插入艺术字时，打开菜单栏中的“插入”菜单，并选择图 2-42 所示的“图片”二级菜单中的“艺术字”命令，此时将出现一个“‘艺术字’库”对话框。其中列出了多种 Word 2000 提供的艺术字式样，从中选定所需的艺术字图形对象的式样后，单击“确定”按钮，此时将出现如图 2-45 所示的“编辑‘艺术字’文字”对话框。在“文字”框中键入要设置为艺术字格式的文字内容，并按照与在“格式”工具栏

中设置文字格式类似的方法设置好文字的字体、字号和字型（是否加粗、是否倾斜），然后单击“确定”按钮即可在文档中插入艺术字。

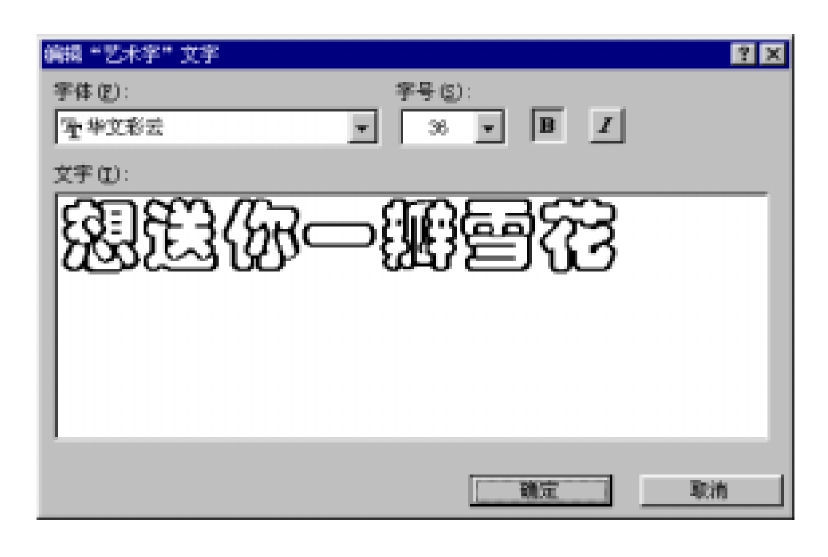


图 2-45　“编辑‘艺术字’文字”对话框

利用 Word 2000 的艺术字功能设计出的艺术字效果如图 2-46 所示。

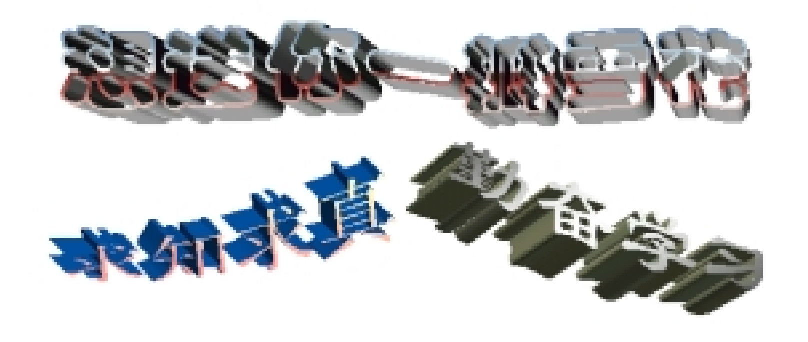


图 2-46　艺术字效果

2.7.5　编辑图片和图形

在文档中插入图片或图形之后，如果用户对其效果不满意，可以对它们进行编辑。

1. 编辑图片

对于插入到文档中的剪贴画或者从其他路径下导入的图片，可以利用如图 2-47 所示的“图片”工具栏进行编辑。如果 Word 2000 中没有显示“图片”工具栏，则可在菜单栏、“常用”或“格式”工具栏的任意位置单击鼠标右键，并从弹出的快捷菜单中选择“图片”命令显示“图片”工具栏。也可在图片上单击鼠标右键并从弹出的快捷菜单中选择“显示‘图片’工具栏”命令。

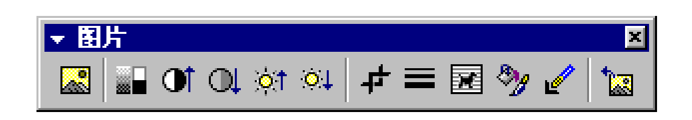


图 2-47　“图片”工具栏

单击“图片”工具栏中的“插入图片”按钮将出现“插入图片”对话框，因此此按钮的作用相当于图 2-42 所示的“图片”二级菜单中的“来自文件”命令，用于在光标处插入图片。

利用“图片”工具栏可以对图片的图像效果进行调整。选定图片后，单击“图像控制”按钮可以从其下拉菜单中为图片选择“自动”、“灰度”、“黑白”及“水印”几种色调效果。连续单击“增加对比度”按钮或“降低对比度”按钮可以调整图片的对比度，连续单击“增加亮度”按钮或“降低亮度”按钮可以调整图片的亮度。图片的不同图像效果如图 2-48 所示。

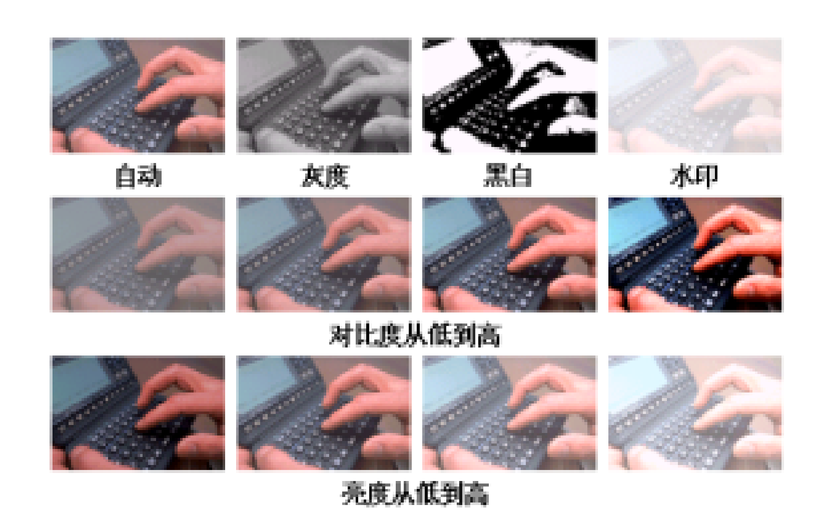


图 2-48　图片的不同图像效果

裁剪图片时，首先选定要裁减的图片，然后单击“图片”工具栏中的“裁剪”按钮，此时鼠标指针将变为裁剪框形状，将指针置于图片任何一个控点（位于图片四角及各边中点，以小方块表示）上，按住左键并拖动鼠标，此时将以虚线框表示裁减后保留的图片范围，调整至所需大小后松开左键即可。

另外，利用“图片”工具栏中的线型按钮可以设置线条粗细，“设置透明色”按钮可以将图片中的实色部分改为透明色。

单击“重设图片”按钮可以使图片恢复为最初的形式。

2. 编辑自选图形

利用如图 2-49 所示的“绘图”工具栏可以对绘制的自选图形进行编辑。如果 Word 2000 中没有显示“绘图”工具栏，则可在菜单栏、“常用”或“格式”工具栏的任意位置单击鼠标右键，并从弹出的快捷菜单中选择“绘图”命令显示“绘图”工具栏。

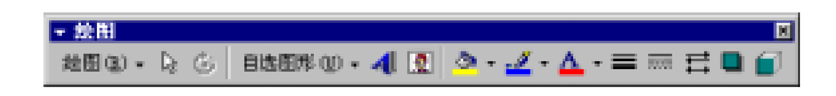


图 2-49　“绘图”工具栏

对已有的自选图形可以改变其形状。首先选定要改变的自选图形，然后选择“绘图”工具栏的“绘图”菜单中的“改变自选图形”命令，并从其子菜单中重新选定所需的自选图形形状。

单击“选择对象”按钮后，按住左键并拖动鼠标可以选定若干个图形对象。

单击“绘图”工具栏中的“自由旋转”按钮后，图形四角将出现绿色的小圆控点，同时鼠标指针变为圆环箭头状，将此指针置于任一控点上，按住左键并拖动鼠标可以将图形旋转任意角度。

“绘图”工具栏中的“自选图形”按钮、“插入艺术字”按钮以及“插入剪贴画”按钮的作用分别相当于图 2-42 所示的“图片”二级菜单中的“自选图形”、“艺术字”以及“剪贴画”命令，这里不再重复介绍。

在绘制出的自选图形中还可以添加文字（线条、连接符或任意多边形除外）。只要单击图形即可键入文字。键入的文字会成为自选图形的一部分。

为了增强图形的色彩效果，选定图形后，单击“填充颜色”按钮、“线条颜色”按钮以及“字体颜色”按钮可以分别为图形内部、线条以及图形中的文字添加颜色。如果想改变预设的色彩，可以单击各按钮右端的下拉箭头并从弹出的调色板中选择合适的颜色或色彩方案。

从“线型”、“虚线线型”按钮的下拉列表中可以指定线条的粗细，从“箭头样式”按钮的下拉列表中可以指定不同的箭头样式。

单击“阴影”按钮或“三维效果”按钮将弹出下拉菜单，从中可以为图形设置阴影效果或三维效果。

利用“绘图”工具栏的强大功能对自选图形进行编辑之后，图形的效果将变得更加丰富多彩。例如，使用“绘图”工具栏对图 2-44 中

绘制的自选图形进行处理后，图形效果如图 2-50 所示。



图 2-50　使用“绘图”工具栏处理后的图形效果

3. 编辑艺术字

作为一种装饰性的文字，艺术字这种特殊的文字效果属于图形对象的范畴，可以使用“绘图”工具栏对其进行处理，方法与编辑自选图形相同。

除此之外，Word 2000 还提供了“艺术字”工具栏专门用于编辑艺术字。在文档中插入艺术字之后，单击具有艺术字效果的文字，将出现如图 2-51 所示的“艺术字”工具栏，利用这个工具栏可以对艺术字进行进一步的设置。如果 Word 2000 中没有显示“艺术字”工具栏，则可在菜单栏、“常用”或“格式”工具栏的任意位置单击鼠标右键，并从弹出的快捷菜单中选择“艺术字”命令显示“艺术字”工具栏。

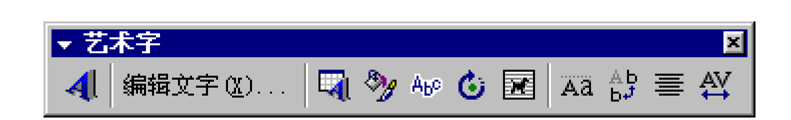


图 2-51 “艺术字”工具栏

“艺术字”工具栏中的“插入艺术字”按钮的作用相当于图 2-42 所示的“图片”二级菜单中的“艺术字”命令，用于创建艺术字。

Word 2000 允许对已有的艺术字再次进行编辑。单击“艺术字”工具栏中的“编辑文字”按钮将再次打开图 2-45 所示的“编辑‘艺术字’文字”对话框，从而可以重新设置文字内容及其字体、字号和字型。也可通过双击要更改的艺术字图形对象来打开“编辑‘艺术字’文字”对话框。

如果要更改艺术字的式样，可以在选定艺术字图形对象后单击“艺术字”工具栏中的“艺术字库”按钮打开“‘艺术字’库”对话框。从中选定新的艺术字式样后，单击“确定”按钮即可更改艺术字的式样。

单击“艺术字”工具栏中的“艺术字形状”按钮可从其下拉列表中选定艺术字图形对象的形状。

另外，利用“艺术字”工具栏中的按钮还可以对艺术字图形对象进行旋转、将艺术字设为竖排文字、更改艺术字间距等处理。

2.7.6 设置图片和图形的格式

Word 2000 提供专门的对话框来设置图片和图形的格式。利用“设置图片格式”对话框、“设置自选图形格式”对话框以及“设置艺术字格式”对话框可以分别对图片、自选图形以及艺术字图形对象的格式进行设置。

由于“设置图片格式”对话框、“设置自选图形格式”对话框以及

“设置艺术字格式”对话框三者的结构与用法都非常类似，因此这里将它们放在一起统称为“设置格式”对话框进行介绍，并注意强调了有区别的地方。

1. 打开“设置格式”对话框的方法

打开“设置图片格式”对话框的方法有：

（1）双击文档中的图片。

（2）在图片上单击鼠标右键并从弹出的快捷菜单中选择“设置图片格式”命令。

（3）单击图 2-47 所示的“图片”工具栏中的“设置图片格式”按钮。

（4）选定图片，然后打开菜单栏中的“格式”菜单，并选择其中的“图片”命令。

打开“设置自选图形格式”对话框的方法有：

（1）双击文档中的自选图形。

（2）在自选图形上单击鼠标右键并从弹出的快捷菜单选择“设置自选图形格式”命令。

（3）选定自选图形，打开菜单栏中的“格式”菜单并选择其中的“自选图形”命令。

打开“设置艺术字格式”对话框的方法有：

（1）在艺术字上单击鼠标右键并从弹出的快捷菜单中选择“设置艺术字格式”命令。

（2）单击图 2-51 所示的“艺术字”工具栏中的“设置艺术字格式”按钮。

（3）选定艺术字对象，打开菜单栏中的“格式”菜单并选择其中的“艺术字”命令。

2. 颜色和线条设置

单击“设置格式”对话框中的“颜色和线条”标签可切换到如图 2-52 所示的“颜色和线条”选项卡。从中可以对图片或图形的填充色、线条及箭头进行设置。



图 2-52 “颜色和线条”选项卡

在“填充”栏的“颜色”框里可以指定图片或图形的填充色，选中“半透明”复选框可将所选的实色填充设为半透明，清除该复选框时所选的填充设为不透明。如果所选对象包含过渡、纹理、图案或图片填充，则该选项无效。

在“线条”栏中可以设置线条的颜色、虚实、线型及粗细。其中线条的颜色、虚实和线型可从相应框的下拉列表中选取，线条的粗细可在“粗细”框中选定或直接键入。

如果图案中包括了箭头，则可在“箭头”栏中指定箭头前端、后端的形状及大小。

✍ 对于某个特定的的图片或图形而言，“颜色和线条”选项卡中的“半透明”复选框、“线条”栏及“箭头”栏的某些选项甚至整栏，可能是不可选的。

3. 大小设置

单击“设置格式”对话框中的“大小”标签可切换到如图 2-53 所示的“大小”选项卡，从中可以对图片或图形的尺寸及旋转进行设置。

设置图片或图形的大小时，可以在“尺寸和旋转”栏的“高度”和“宽度”框中选定或键入所选对象的高度和宽度的数值，也可在“缩放”栏的“高度”和“宽度”框中选定或键入以原始尺寸的百分比所表示的高度和宽度。如果选中“锁定纵横比”复选框，那么所选图案的高度与宽度将受到限制，“高度”框或“宽度”框中的值将会随着另外一个设置的变化而变化，以使其相互保持原始比例。

对于自选图形和艺术字图形对象，可在“旋转”框中选定或键入所选对象按顺时针方向旋转的度数（从 0º 到 359º）。

✍ “高度”框和“宽度”框中的数值都是针对对象未旋转时

的数据而言。

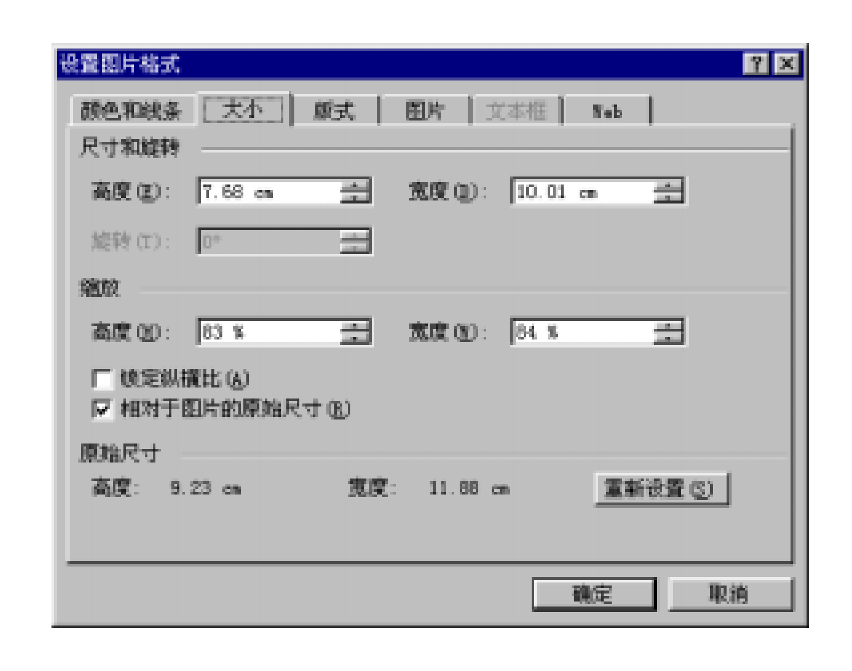


图 2-53　“大小”选项卡

对于图片对象，在“原始尺寸”栏中显示了图片的原始高度和宽度。若选中“缩放”栏中的“相对于图片的原始尺寸”复选框，则将根据图片的原始尺寸计算“缩放”栏“高度”框和“宽度”框中的百分比。单击“重新设置”按钮可以将所选图片的高度和宽度还原为原始大小。

✍ 对于图片而言，“大小”选项卡中的“旋转”框是不可选的；而对于自选图形以及艺术字图形对象而言，“缩放”栏中的“相对于图片的原始尺寸”复选框是不可选的，同时“原始尺寸”栏也无效。

除了可以在“大小”选项卡中设置图片和图形的尺寸外，还可以

直接用鼠标来调整对象的大小。选定要调整大小的图片或图形对象后，所选对象的四角以及四条边的中点将出现八个尺寸控点，拖动这些尺寸控点可以将图片或图形调整到所需的形状和大小。这种方法比在“大小”选项卡中设置尺寸更方便、直观，缺点是不够精确。

前面也已经介绍过，对于自选图形或艺术字图形对象，除了在“大小”选项卡中设置旋转角度外，还可以利用“绘图”工具栏或“艺术字”工具栏中的“自由旋转”按钮进行旋转。

4. 版式设置

单击“设置格式”对话框中的“版式”标签可切换到如图 2-54 所示的“版式”选项卡。从中可以对图片或图形的文字环绕方式以及水平对齐方式等进行设置。

“环绕方式”栏中共包括了五种文字环绕方式。其中“嵌入型”使对象与文字处于同一层，随文字的移动而移动；“四周型”使文字环绕在所选对象的边界框四周；“紧密型”使文字紧密环绕在图案自身的边缘（而不是对象边界框）的周围；“浮于文字上方”使对象处于文档中文字的上层并覆盖下层的文字，对象不随文字的移动而移动；“衬于文字下方”使对象处于文档中文字的下层，对象不随文字的移动而移动。



图 2-54　“版式”选项卡

在文档中插入图片或图形时，Word 2000 默认剪贴画以“嵌入型”的文字环绕方式插入，而自选图形及艺术字图形对象以“浮于文字上方”的文字环绕方式插入。若想更改所选对象的文字环绕方式，只需单击“环绕方式”栏中的相应选项。

在“水平对齐方式”栏中可以设置图片或图形的水平对齐方式，包括“左对齐”、“居中”、“右对齐”和“其他方式”等几个选项。“左对齐”使对象的左边缘与文档的左页边距对齐，“居中”使将对象的中心与文档的中心对齐；“右对齐”使对象的右边缘与文档的右页边距对齐；选中“其他方式”后，可以根据指定的度量方式对齐对象（单击“高级”按钮并在弹出的“高级版式”对话框的“图片位置”选项卡中指定用于对齐对象的绝对位置）。

- 在“高级版式”对话框中还可以设置垂直对齐的选项以及其他的文字环绕方式（如穿越型、上下型）。
- 对于自选图形而言，是不能为其设置“嵌入型”文字环绕方式的。而对于图片及艺术字图形对象而言，一旦为其设置了“嵌入型”文字环绕方式，则“水平对齐方式”栏当然也就不可选了。

5. 在“图片”选项卡中的设置

单击“设置格式”对话框中的“图片”标签可切换到如图 2-55 所示的“图片”选项卡，从中可以对图片的裁剪及图像效果进行设置。

- “图片”选项卡是专门针对图片对象进行格式设置的。对于自选图形与艺术字图形对象而言，“格式设置对话框”中的“图片”选项卡是不可选的。
- 在“图片”选项卡中进行的设置完全可以按前面介绍过方法利用图 2-47 所示的“图片”工具栏中的相应按钮来实现，这种方法比在“图片”选项卡中进行设置更方便、直观，缺点是不能够指定精确的数值。



图 2-55　“图片”选项卡

2.8　边框和底纹

边框和底纹可以有效地美化文档版面的外观。在 Word 2000 中可以给页面、文字、图片以及表格添加边框，并可用底纹填充文字及表格的背景。既可以利用“表格和边框”工具栏来添加边框和底纹，也可在“边框和底纹”对话框中进行。

2.8.1　利用“表格和边框”工具栏添加边框和底纹

在菜单栏、“常用”工具栏或“格式”工具栏的任意位置单击鼠标右键，并从弹出的快捷菜单中选择“表格和边框”命令将显示出如图 2-56 所示的“表格和边框”工具栏。利用这个工具栏可以完成插入表格、绘制表格、单元格合并及拆分等操作。除此之外，“表格和边框”

工具栏还能用来添加边框和底纹。

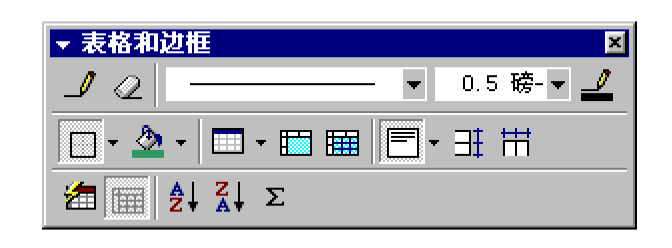
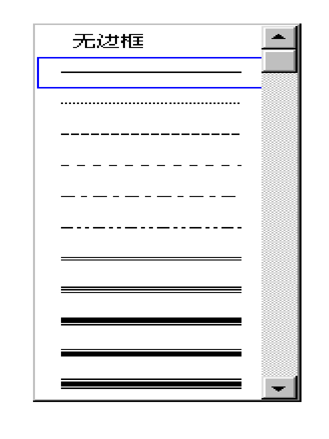


图 2-56 “表格和边框”工具栏

1. 添加边框

利用图 2-56 所示的“表格和边框”工具栏可以为选定的文字、图片及表格添加边框。

添加边框时，首先应当选定要添加边框的对象（文字、图片、整张表格或表格的一部分），然后单击工具栏中“线型”框右端的下拉箭头，并从如图 2-57 所示的下拉列表中为要添加的边框选择合适的线型。类似地，单击“粗细”框右端的下拉箭头并在如图 2-58 所示的下拉列表中确定框线的粗细。



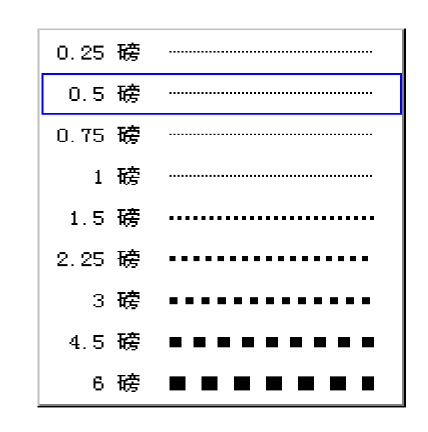


图 2-57 “线型”下拉列表　　　　图 2-58 “粗细”

下拉列表

设置好框线的线型及线条粗细后，单击“边框颜色”按钮按并从弹出的调色板中选择线条的颜色。

最后还应当选择要添加的边框的类型。单击“边框”按钮右端的下拉箭头将出现如图 2-59 所示的下拉列表，其中包括了各种边框的类型。按图中的顺序依次为：“外部框线”、“所有框线”、“上框线”、“左框线”、“内部横框线”、“斜下框线”、“横线”、“内部框线”、“无框线”、“下框线”、“右框线”、“内部竖框线”和“斜上框线”。单击某种类型可以为选定对象添加相应的框线。

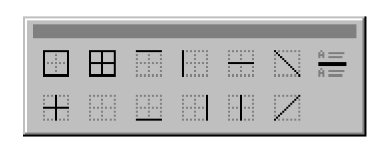


图 2-59　“边框”下拉列表

- 上述各种框线类型中，“斜上框线”和“斜下框线”只适用于表格。
- 如果选定了多个段落，则 Word 2000 在添加框线的时候是以段落为基本单位的（而不是行）。例如，为多个段落添加“内部横框线”时，将在各个段落之间而不是各行之间添加横框线。
- “无框线”可用于删除所选对象的所有边框。对表格而言，如果删除了边框，在单元格边界处仍然可能看到虚框（但打印时不会被打印），单击“表格和边框”工具栏上的“隐藏/显示虚框”按钮可隐藏或显示虚框。

2. 添加底纹

利用图 2-56 所示的“表格和边框”工具栏可以为选定的文字、表格添加底纹。

添加底纹时，首先选定要添加底纹的对象（文字、整张表格或表格的一部分），然后单击“表格和边框”工具栏中的“底纹颜色”按钮右端的下拉箭头，并从弹出的调色板中单击所需的底纹颜色即可。

若想删除已经添加的底纹，首先选定要删除底纹的对象，然后在上述调色板中选中“无填充颜色”选项即可。

2.8.2 利用“边框和底纹”对话框添加边框和底纹

打开菜单栏中的“格式”菜单，并选择其中的“边框和底纹”命令，将出现如图 2-60 所示的“边框和底纹”对话框。利用这个对话框，可以给页面、文字、图片以及表格添加边框，给文字及表格添加底纹，同时还可以对边框和底纹的格式进行比在“表格和边框”工具栏中更全面的设置。

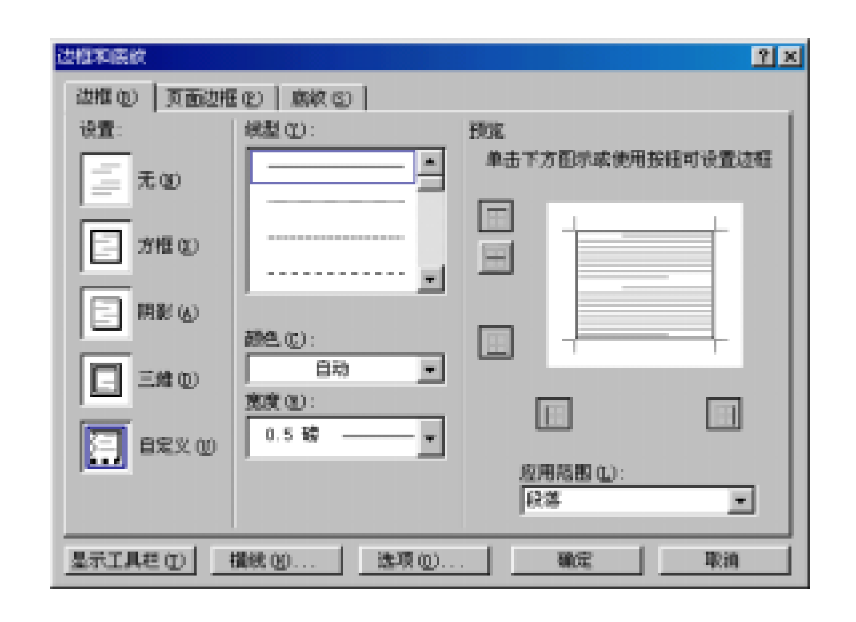


图 2-60　“边框和底纹”对话框中的“边框”选项卡

“边框和底纹”对话框包括“边框”、“页面边框”以及“底纹”三个选项卡，以下将分别进行介绍。

✍ 当选定的对象是图片时，由于无法对图片添加底纹，所以此时的对话框为“边框”对话框，同时“底纹”选项卡为不可选。

1. “边框”选项卡

单击“边框和底纹”对话框中的“边框”标签可以切换到图 2-60 所示的“边框”选项卡，在这个选项卡中可以给文字、图片以及表格添加边框。

在“边框”选项卡的右下角有一个“应用范围”框，单击“应用范围”框右端的下拉箭头将出现下拉列表，根据选定对象的不同，下

拉列表中可能包括了“文字”、“段落”、“图片”、“单元格”、“表格”等选项。从中选定应用边框格式的范围。

在“设置”栏中可以选择边框的格式。当在“应用范围”框中选定“文字”、“段落”或者“图片”时，“设置”栏中包括“无”、“方框”、“阴影”、“三维”及“自定义”等选项；当在“应用范围”框中选定“表格”或“单元格”时，“设置”栏中包括“无”、“方框”“全部”“网格”“自定义”等选项。单击所需的边框格式。

- 若在“设置”栏中选中“无”可删除选定对象的所有边框。
- 在“设置”栏中选中“自定义”表示用在“预览”图示中单击的选项创建自定义边框。如果在“预览”图示中单击了某边或边框按钮，Word 2000 将自动选中“自定义”选项。

与在“表格和边框”工具栏中进行设置类似，在“线型”、“颜色”和“宽度”框中选定框线的线型、颜色及宽度。

在“预览”图示中可以预览在“边框”选项卡中设置的边框效果。除此之外，还可以单击模型的边或边框按钮创建自定义边框格式。

完成对边框的设置后，单击“确定”按钮即可。

- 如果在“应用范围”框里选中了“段落”选项，可以进一步对段落边框的位置进行设置。单击“选项”按钮将出现一个“边框和底纹选项”对话框。在“距正文”栏的“上”、“下”、“左”、“右”框里可以分别选定或键入段落边框的

上、下、左、右框线与正文文字之间的距离，在“预览”栏中可以预览设置效果。完成设置后，单击“确定”按钮返回“边框”选项卡。

2. “页面边框”选项卡

单击“边框和底纹”对话框中的“页面边框”标签可以切换到如图 2-61 所示的“页面边框”选项卡。在这个选项卡中可以给页面添加边框。



图 2-61　“边框和底纹”对话框中的“页面边框”选项卡

可以看出，“页面边框”选项卡与“边框”选项卡的外观非常类似，二者的用法也基本相同。

单击“应用范围”框右端的下拉箭头将出现下拉列表，其中包括“整篇文档”、“本节”、“本节-只有首页”以及“本节-除首页外所有

页”等选项。从中选定添加页面边框的范围。

在“设置”栏中包括的“无”、“方框”、“阴影”、“三维”及“自定义”等选项中选定所需的页面边框格式。

✍ 在“设置”栏中选中“无”可删除“应用范围”框中指定对象的页面边框。

在“线型”、“颜色”和“宽度”框中选定页面边框框线的线型、颜色及宽度。

若有必要，可在“艺术型”框的下拉列表中选择图形式页面边框，选定的艺术型边框将替换在“线型”框中进行的设置。

在“预览”图示中可以预览在“页面边框”选项卡中设置的页面边框效果。除此之外，还可以单击模型的边或边框按钮创建自定义页面边框格式。

如果要对页面边框的精确位置等属性进行进一步的设置，可单击“选项”按钮，此时将出现如图 2-62 所示的“边框和底纹选项”对话框。首先确定度量标准。在“度量标准”框的下拉列表中包括“文字”和“页边”两个选项。选中“文字”可以页边距为基准设置页面边框的内侧边界；选中“页边”可以页边距为基准设置页面边框的外侧边界。以“文字”为度量标准时，在“边距”栏的“上”、“下”、“左”、“右”框里可以分别选定或键入页面边框内侧与正文边界之间的距离；若以“页边”为度量标准，则指定的应是页面每边与页面边框外侧之

间的距离。在“选项”栏中为页面边框的格式选定所需的复选框。在“预览”栏中可以预览设置效果。完成设置后，单击“确定”按钮返回“边框”选项卡。

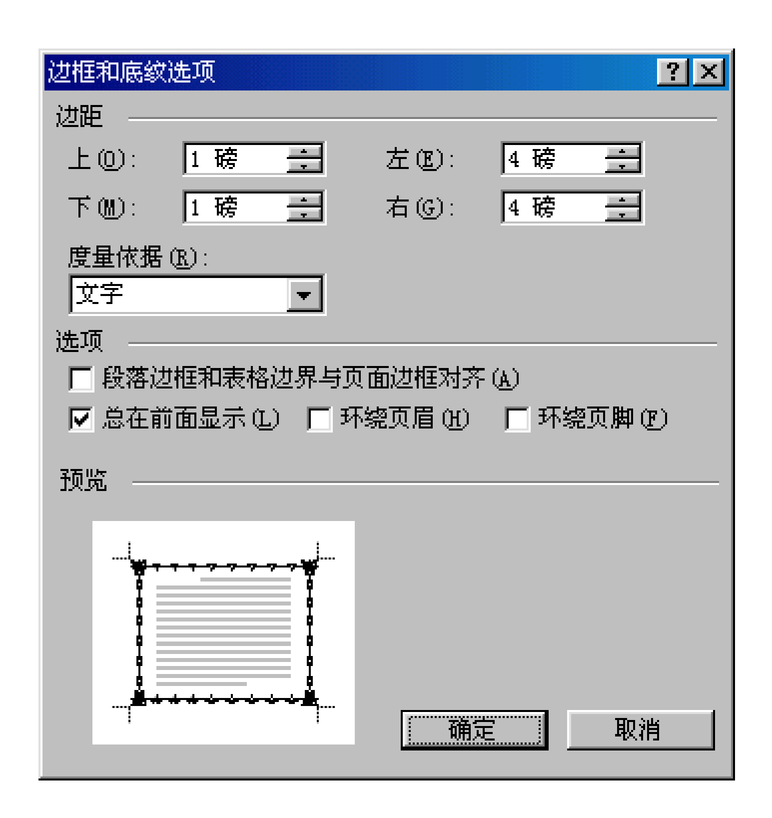


图 2-62 “边框和底纹选项”对话框

完成对页面边框的设置后，单击“确定”按钮即可。

3. “底纹”选项卡

单击“边框和底纹”对话框中的“底纹”标签可以切换到“底纹”选项卡，如图 2-63 所示。在这个选项卡中可以给文字及表格添加底纹。

单击“底纹”选项卡的右下角的“应用范围”框右端的下拉箭头将出现下拉列表，根据选定对象的不同，下拉列表中可能包括了“文

字”、“段落”、“单元格”、“表格”等选项。从中选定要添加底纹的范围。

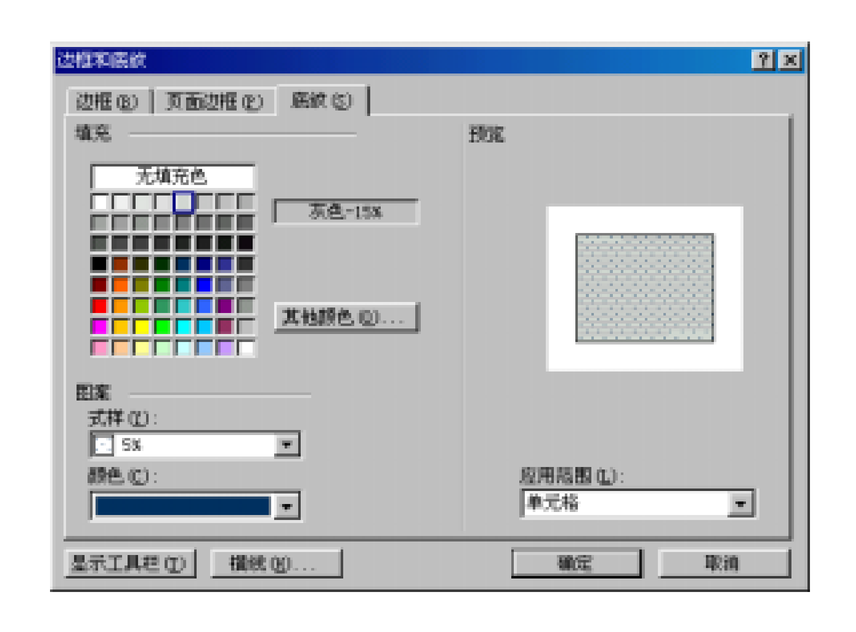


图 2-63　“边框和底纹”对话框中的“底纹”选项卡

在“填充”栏中可以选择底纹的填充色，选中“无填充色”可以取消底纹颜色。

单击“图案”栏的“式样”框右端的下拉箭头将出现下拉列表，从中可以选择底纹式样，所选的底纹式样将应用于“填充”栏中选定颜色的上层。选择“清除”表示只应用填充颜色（无图案颜色）；选择“纯色（100%）”表示只应用图案颜色（无填充颜色）。在“颜色”框中可以选定所选底纹式样图案中线条和点的颜色。

在“预览”图示中可以预览在“底纹”选项卡中设置的底纹效果。

完成对底纹的设置后，单击“确定”按钮即可。

- 单击“边框和底纹”对话框中的“显示工具栏”按钮将关闭对话框并显示“表格和边框”工具栏，此时可利用工具栏中的相应按钮为选定的对象添加边框和底纹。

第三章　更高效地使用 Word 2000

Word 2000 提供了许多高效率的工具。对于一个 Word 2000 的高级用户来说，学习这些知识是很有必要的。

在本章中，将介绍一些更高效地使用 Word 2000 的方法。例如域和宏的使用、使用拼写和语法检查功能、使用 Word 2000 的自动功能等。掌握了这些方法之后，用户可以使自己的工作效率大为提高。

3.1　域

在本节和下一节中，将分别介绍 Word2000 的两项强大功能：域和宏。它们实现了对 Word 文档的自动化操作。

3.1.1　域的概述

域相当于文档中可能发生变化的数据或邮件合并文档中套用信函、标签中的占位符。它具有很强的灵活性，能够控制在文档中插入的文字、图表、索引、目录、图形、编号等等。域的最大的特点就是能够根据文档的改动和其他有关因素的变化来自动更新域中显示的内容。

域虽然属于 Word 中比较抽象、高级的部分，但实际上大部分 Word

2000 用户都曾经不自觉地使用过域的功能。例如，当用户为文档添加页码时，实际上就是在页眉或页脚中加入了一个“PAGE”域。

域有两种表现形式：域代码和域结果。通常在文档中看到的是域的结果。从这个方面来说，域有些类似于数学公式。域代码是公式本身，而域结果则是代入具体数值以后公式计算得到的结果。

域可以划分为以下三种类型：

（1）结果域：大部分的域都是结果域，它用于在文档中显示计算得到的信息。

（2）行为域：用于对文档进行某些操作，但不在文档中显示可见的信息。

（3）标记域：既不在文档中显示信息，也不对文档进行某项操作，只是简单地标记文档中某部分的位置。

3.1.2 域的组成

域一般由三个部分组成：特征字符、域类型以及域开关。例如：{ TIME \@ "yy-M-d" }，下面将以这个域为例对域的各部分进行介绍。

1. 特征字符

域的特征字符就是花括弧{ }。它们可以通过组合键“Ctrl+F9”输入得到。这一对花括弧的作用是将括弧中的域代码和括弧外的文档区分开来。

2. 域类型

域类型其实就是域名。不同的域有不同的作用，如上面所示的域，其域名为 TIME，表明这个域将返回当前的时间。

3. 域开关

域开关主要用于改变域的缺省功能或者改变域结果的格式。域开关一般由一个反斜杠“\”开始，后面跟一个简单字符，如上面的“\@”。

不同的域有不同的域开关，但是有一些开关，对几乎所有的域都是适用的，这样的域开关被称为“通用开关”。通用开关主要用于修改显示域结果的格式。

主要的通用开关有：格式开关“*”、数字开关“\#”、日期—时间开关“\@”以及锁定开关“\!”。

格式开关“*”用于指定域结果显示的数字格式、大小写方式以及字符格式；并防止在更新域时修改域结果的格式。其主要的开关选项如表 3-1、表 3-2 及表 3-3 所示。

表 3-1 大小写格式开关选项

开关选项	功能	示例
* Caps	每个单词的首字母大写	**{** Author " * Caps **}** 显示为“Tang Hao”
* FirstCap	第一个单词的首字母大写	**{** QUOTE "word" * FirstCap **}** 显示为“Word”
* Upper	所有字母均大写	**{** COMMENT * UPPER **}**显示为“I BILIEVE I CAN FLY”。

* Lower	所有字母均小写	**{** FILLIN "What’s wrong with you" * Lower **}**显示为“what’s wrong with you”

表 3-2　数字格式开关选项

开关选项	功能	示例
* alphabetic	将结果显示为字母字符。该结果和域代码中的单词“alphabetic”具有相同的大小写。	**{** SEQ appendix * ALPHABETIC **}**显示为“B”（而不是“2”），而**{** SEQ appendix * alphabetic **}** 显示为“b”
* Arabic	结果显示为阿拉伯数字。	**{** PAGE * Arabic **}** 显示为“31”
* CardText	结果显示为基数词。设置为小写格式。	**{** = SUM(A1:B2) * CardText **}** 显示为“seven hundred ninety”，而**{** = SUM(A1:B2) * CardText * Caps **}** 显示为“Seven Hundred Ninety”
* DollarText	结果显示为基数词。Word 在小数点的位置插入“and”并将小数部分显示为以阿拉伯数字表示的前两位小数（舍入后）作为分子、以 100 作分母的分数。结果设置为小写的格式。	**{** = 9.20 + 5.35 * DollarText * Upper **}**显示为“FOURTEEN and 55/100”

续　表

开关选项	功能	示例
* Hex	结果显示为六进制数	**{** QUOTE "458" * Hex **}** 显示为“1CA”
* OrdText	序数词。结果设置为小写格式。	**{** DATE\@"d"*OrdText **}**显示为“twenty-first”，而**{** DATE \@ "d" * OrdText*FirstCap **}** 显示为“Twenty-first”

* Ordinal	结果显示为阿拉伯数字表示的序数词。	**{** DATE \@ "d" * Ordinal **}** 显示为“30th”
* roman	结果显示为罗马数字。结果和域代码中的单词“roman”具有相同的大小写。	**{** SEQ CHAPTER * roman **}** 显示为“xi”，而 **{** SEQ CHAPTER * ROMAN **}** 显示为“XI”

表 3-3　字符格式开关选项

开关选项	功能	示例
* Charformat	将域类型的第一个字母的格式作用于整个域结果。	**{** **R**EF chapter2_title * Charformat **}**显示为“**Whales of the Pacific**”。
* MERGEFORMAT	将以前域结果所使用的格式作用于当前的新结果。	**{** AUTHOR * MERGEFORMAT **}**显示作者的名字，并且如果该名字是以粗体显示的，那么当域因作者姓名改变而更新时，Word 仍保留该粗体格式。

数字开关“\#”指定数字结果的显示方式。该开关之所以称为“图片”，是因为用了符号来代表域结果的格式。例如，域 **{** = SUM(ABOVE) \# $#,##0.00 **}** 中的开关 \# $#,##0.00 可使结果显示为“$4,455.70”。如果域的结果不是数字，那么该开关不起作用。数字-图片域开关选项如表 3-4 所示。

表 3-4　数字－图片域开关选项

开关选项	功能	示例
0	指定结果中必须显示的数字位数。如果在结果中该位置没有数字，那么 Word	**{** = 4 + 5 \# 00.00 **}** 显示为“09.00”

	显示 0。	
#	指定结果中必须显示的数字位数。如果在结果中该位置没有数字，那么 Word 显示空格。	**{** = 9 + 6 \# $### **}** 显示为“$ 15”
X	省略占位符“x”以左的数字。如果占位符在小数点以后，那么 Word 将结果舍入到该数位。	**{** = 111053 + 111439 \# x## **}** 显示为“492”；**{** = 1/8 \# 0.00x **}** 显示为“0.125”
.（小数点）	确定小数点的位置。	{ = SUM(ABOVE) \# $###.00 } 显示为“$495.47”
,（数字分组符）	每三个数字间用此分隔符分开。	**{** = NetProfit \# $#,###,### **}** 显示为“$2,456,800”
-（减号）	在负结果前添加一个负号，在正或为 0 的结果前添加一个空格。	**{** = 10 - 90 \# -## **}** 显示为“-80”
+（加号）	在为正的结果前添加一个正号，在为负的结果前添加一个负号，而在为 0 结果前添加一个空格。	**{** = 100 - 90 \# +## **}**显示为“+10”，
%、$、* 等等	在结果中添加指定的字符。	**{** = netprofit \# "##%" **}** 显示为“33%”

续　表

开关选项	功能	示例
"positive; negative"	为正、负结果指定不同的数字格式。	**{** Sales95 \# "$#,##0.00;**-$#,##0.00**" **}** 用一般的格式显示该值，如“$1,245.65”
"positive; negative; zero"	为正、负和零结果指定不同的数字格式。	**{** Sales95 \# "$#,##0.00;($#,##0.00); **$0**" **}** 分别将正值、负值和零显示为：$1,245.65, ($ 345.56), **$0**

'text'	在结果中添加文本。用单引号将该文本括起来。	{ = { Price } *8.1% \# "$##0.00 'is sales tax' " } 显示为 " $347.44 is sales tax"

日期－时间开关"\@"用于指定日期或时间的显示方式。该开关之所以称为"图片"，是因为用了符号来代表域结果的格式。例如，域 { DATE \@ "dddd, MMMM d, yyyy" } 中的开关 @ "dddd MMMM d, yyyy" 规定了将日期显示为"Friday, November 10, 1995"。

日期－时间开关包括日期指令、时间指令、A.M./P.M. 及其他文本和标点，如表 3-5、表 3-6 及表 3-7 所示。

表 3-5　日期指令

图片项	功能
Yy	将年份显示为两位数字，对于 01 到 09 的年份前面带 0
Yyyy	将年份显示为四位数字
M	显示月份，对单数字的月份不在其数字前补 0
MM	显示月份，对单数字的月份在其数字前补 0
MMM	将月份显示为其英文拼写前三个字母的缩写
MMMM	显示月份的全称
D	显示日期，对单数字的日不在其数字前补 0
dd	显示日期，对单数字的日在其数字前补 0
ddd	将日期显示为星期中一天的英文拼写的三个字母的缩写
Dddd	将日期显示为星期中一天的全称

表 3-6　时间指令

图片项	功能
h 或 H	显示小时数，对一位数字的小时不在该数字前

	补 0
Hh 或 HH	显示小时数，对一位数字的小时在该数字前补 0
M	显示分钟数，对一位数字的分钟不在该数字前补 0
Mm	显示分钟数，一位数字的分钟在该数字前补 0

表 3-7 A.M/P.M 及其他文本和标点

图片项	功能	示例
AM/PM	将 AM/PM 大写	**{** TIME \@ "h AM/PM" **}** 显示为“9 AM”或“5 PM”
Am/pm	将 am/pm 小写	**{** TIME \@ "h am/pm" **}**显示为“9 am”或“5 pm
A/P	将 AM/PM 显示为大写的缩写	**{** TIME \@ "h A/P" **}**显示为“9 A”或“5 P”

续 表

图片项	功能	示例
a/p	将 am/pm 显示为小写的缩写	**{** DATE \@ "h a/p" **}**显示为“9 a”或“5 p”
'text'	显示用单引号括起来的任意指定的文本	**{** TIME \@ "HH:mm 'clock' " **}** 显示为 “ 12:45 clock”
character	显示指定字符，如 :(冒号)、 - (连字符)、* (星号) 或空格之类	**{** DATE\@ "HH:mm MMM-d, 'yy" **}** 显示为“11:15 Nov-6, '95”

锁定开关“\!”用于防止更新包含在 BOOKMARK、INCLUDETEXT 或 REF 域结果中的域，原先位置的域结果发生了变化。如果没有该开关，Word 2000 将在更新 BOOKMARK、INCLUDETEXT 或 REF 域时

更新包含在这些域结果中的域。

例如：文档 A 中有一个域 **{** INCLUDETEXT C:\\MY DOCUMENTS\B.DOC \! **}**，其作用是在文档 A 中显示文档 B 的内容。而文档 B 中含有一个“PAGE”域。如果更新 INCLUDETEXT 域，那么“\!”开关能防止 Word 2000 更新包含文本中的“PAGE”域，除非文档 B 先发生了更新。该开关确保了 INCLUDETEXT 域插入文档 A 的文本与文档 B 保持一致。要更新“PAGE”域，需先更新文档 B 中的域，然后更新 INCLUDETEXT 域。

3.1.3 插入域

在文档中插入一个域有三种方法：第一种方法是通过“插入”菜单的“域”选项来插入一个域；第二种方法是向文档中直接插入域代码；第三种方法是几种特殊的域可以直接从菜单中插入。下面将分别介绍这几种方法。

1. 使用插入菜单来插入一个域

Word2000 提供了大量的域，而且某些域还可以接收指令。对于初学者来说，一下子记住这么多域和指令几乎是不可能的，因此使用“插入”菜单来插入一个域是最简单，也是最常用的方法。其步骤如下：

（1）单击文本选择域的插入位置。

（2）打开菜单栏中的“插入”菜单，并选择其中的“域”命令，

将出现如图 3-1 所示的“域”对话框。

（3）在“类别”列表中选择要插入域的类别，在“域名”列表中选择要插入的域。例如，插入一个“DATE”域，其类别是“日期与时间”。

（4）在“域”对话框中，“域代码”栏显示了要插入域的代码的格式，“域代码”下面的文本框显示了要插入域的代码，而“说明”栏显示了对该域的简要介绍。



图 3-1　“域”对话框

（5）单击“域”对话框中的“选项”按钮，将出现如图 3-2 所示的“域选项”对话框，利用这个对话框可以为域添加一些开关或格式。

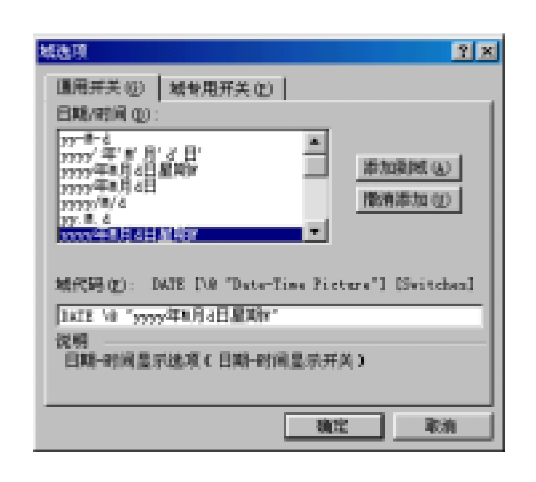


图 3-2　“域选项”对话框

（1）可以在“通用开关”和“域专用开关”中为域选择合适的开关和格式。从列表框中选择需要的开关，然后用鼠标单击“添加到域”按钮即可将其添加到要插入的域中；单击“撤销添加”按钮可将已经添加的开关删除。可以通过对话框的“说明”栏了解各个开关的格式和用法。例如图 3-2 中为“DATE”域添加一个“yyyy 年 m 月 d 日星期 w”的开关，这个开关是用于控制所要显示日期的格式的。

（2）单击“确定”按钮，返回“域”对话框。

（3）单击“确定”按钮，即可在插入点的位置插入“DATE”域。

2. 在文档中直接插入域代码

用菜单插入域虽然简单，但其步骤比较繁琐。如果用户对 Word2000 中的域比较熟悉的话，还有一种更简便的方法来插入域，即在文档中直接插入域代码，其步骤如下：

（1）单击文档选择域的插入点。

（2）按下组合键“CTRL+F9”，此时插入点出会出现带有灰色底纹的花括弧{ }。

（3）在花括弧中输入域的代码（包括域名和开关）。例如在其中输入“DocProperty “Author””，然后再按下 F9 键，即可在域中显示出文档的作者名。

✍ 向文档中直接插入域代码时，其中的花括弧{ }必须是通过输入组合键“Ctrl+F9”得到的，而不能直接通过键盘上的符号键输入。否则 Word 2000 不会将花括弧中的内容视为域代码。

3.1.4 常用的域操作

常用的域操作包括域代码与域结果之间的切换、域底纹的显示、域的格式及样式编排、域的更新与锁定、解除域连接等等。以下将分别进行介绍。

1. 域代码和域结果的切换

一般情况下，在文档中看到的是域的结果。有时用户需要查看域的代码，此时就要在域代码和域结果之间进行切换。



图 3-3　“域”快捷菜单

切换单个域的显示方式时，首先单击域，然后在域上单击鼠标右键，此时将弹出如图 3-3 所示的快捷菜单，选择其中的“切换域代码”命令即可。

另外，单击要切换的域以后，按下“Shift＋F9”组合键也可以在域代码和域结果之间相互切换。

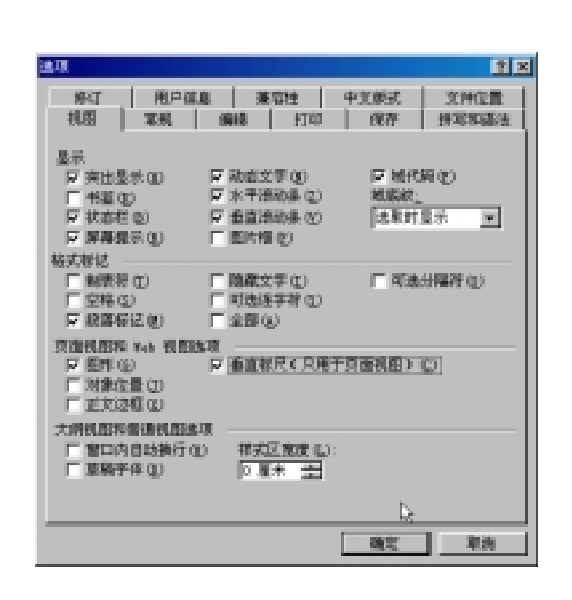


图 3-4　“选项”对话框中的“视图”选项卡

切换文档中所有域的显示方式时，打开菜单栏中的“工具”菜单，并选择其中的“选项”命令，将出现一个“选项”对话框。单击该对

话框中的“视图”标签可切换到如图 3-4 所示的“视图”选项卡。在“显示”栏有一个“域代码”复选框。选中此复选框，将会在文档中显示所有域的域代码；清除此复选框，则文档将会显示所有域的域结果。

另外，按下“Alt＋F9”组合键也可以对文档中的所有域进行切换。

同样，在打印文档时，也可以打印出域的结果而不是域代码。打开菜单栏中的“工具”菜单，并选择其中的“选项”命令，将出现一个“选项”对话框。单击该该对话框中的“打印”标签可切换到如图 3-5 所示的“视图”选项卡。可以看到，在“打印文档的附加信息”栏中有一个“域代码”复选框。选中此复选框，将打印时会打印出所有域的域代码；清除此复选框，则打印时将打印出所有域的域结果。

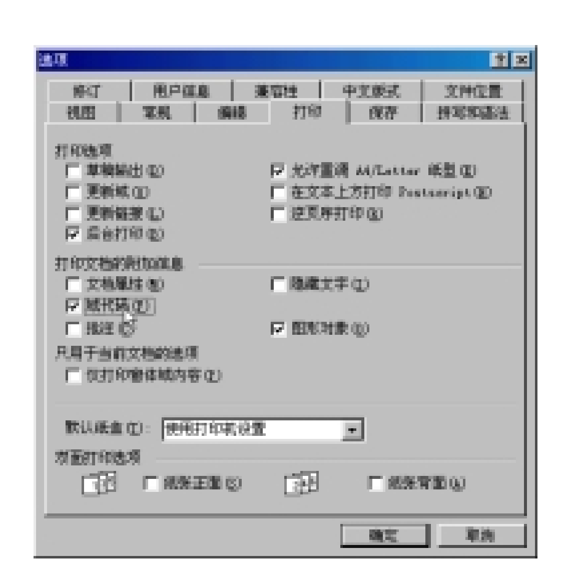


图 3-5　“选项”工具栏的“打印”页面

✍ 在打印文档之前，可以选中“打印选项”中的“更新域”

复选框，将文档中的所有域更新一遍。这样可以保证我们打印出的域的结果的正确性。

2. 域底纹的显示

在缺省的情况下，选定一个域时，Word2000 会自动给域添加灰色的底纹，以将域和文档的其他文本区分开来。

用户可以自己选择域底纹的显示方式。按照前面的方法打开图 3-4 所示“选项”对话框并切换到“视图”选项卡，在“域底纹”框的下拉列表框中可以选择合适的选项，如“不显示”、“显示”、“选择时显示”等等。完成设置后单击“确定”按钮即可。

✍ 可以在文档中看到域底纹，但是在打印时，域底纹是不会被打印出来的。

3. 域的格式和样式

同文档中的其他内容一样，可以对域应用任何样式和格式。例如可以将一个域的“样式”名设置为“标题 1”，可以将域的字体设置为“楷体”，字号设置为“五号”，还可以设置它的前景色和背景色等等。

如果用户对域代码应用了某种格式和样式，则域结果也将显示为相应的格式和样式，反之亦然。

4. 域的更新

域的一个最大的好处就是它能够在文档中快速地插入一些信息，并且这些信息是动态的。也就是说，这些域的内容能够随着外部环境

或者文档内容的变化自动更新，这就省去了大量的修改文档的繁琐工作。

使用鼠标或者键盘都可以更新域。

使用鼠标更新一个域时，用鼠标单击要更新的域，使光标位于该域上，然后单击鼠标右键，将会弹出如图 3-6 所示的快捷菜单，选择快捷菜单中的“更新域”命令，即可将选中的域更新。



图 3-6　使用快捷菜单更新域

使用键盘更新域时，首先将光标移动到要更新的域上，然后按下 F9 键，即可将选择的域更新。

✍ 如果想要同时更新文档中的所有域，可以先选中全文，再用上面介绍的方法来更新域。

5. 域的锁定

使用域的一大好处是能够向文档中插入动态的信息。但是，有时候用户希望保持当前的域结果，而不希望再对域进行更新。这时可以选择两种办法来达到这个目的：锁定域和解除域的连接。

首先介绍锁定一个域和解除一个域的锁定的方法。

如果锁定了一个域，那么在解除域的锁定之前，将不能对这个域进行更新，即当前的域结果将保持不变。

锁定一个域时，首先将光标放置在要锁定的域上，然后按下“Ctrl+F11”组合键，即可将该域锁定。

如果想要更新锁定了的域，必须先解除域的锁定。解除对域的锁定时，首先将光标放置在要解除锁定的域上，然后按下“Ctrl+Shift+F11”组合键，即可解除对域的锁定。

6. 解除域的连接

解除域的连接，就是将当前的域结果变成常规文本，并且不能再将此常规文本在恢复成域。如果需要更新信息的话，必须再次插入此域。

解除一个域的连接时，首先单击该域，然后按下“Ctrl+Shift+F9”组合键即可。

3.1.5 常用域

想要成为Word2000的高级用户，熟练地掌握一些常用的域是必不可少的。这将给用户带来巨大的便利。下面将介绍在Word2000中最常用的域（按字母顺序排列）。

1. Ask 域

Ask 域的语法为：

{ ASK *Bookmark* **"***Prompt***"** [*Switches*] **}**

Ask 域的指令和开关如表 3-8 所示。

表 3-8　Ask 域的指令和开关

指令或开关	解释
Bookmark	指定给输入信息的书签名。例如“姓名”。
"Prompt"	显示在对话框中的提示文字，例如“作者姓名：”。
\d “缺省值”	指定用户没有在提示对话框中键入应答信息时，使用的默认应答信息。例如，如不键入应答信息，那么域{ ASK 姓名“输入打字员姓名的缩写” \d “唐浩” }将“唐浩”分配给书签“姓名”。如不指定默认应答信息，Word 将使用最后输入的应答信息。如果要指定空白的默认应答信息，则应在该开关后面键入空的引号对（\d ""）。
\o	指定了在邮件合并主文档中使用该域时，只显示一次提示信息，而不是每次合并新的数据记录都显示提示。在每篇合并结果文档中插入相同的响应信息。

Ask 域的作用是：提示输入信息并指定一个书签来代表输入的信息。必须在文档中将一个 REF 或 BOOKMARK 域插在 ASK 域之后，Word 2000 才会在此打印输入的信息。

Word 2000 将在每次更新 ASK 域时显示提示信息。在输入新的应答信息之前，原有信息会一直保留在书签中。如果在邮件合并主文档中使用 ASK 域，那么每次合并新记录时都会显示提示信息（除非使用 \o 开关）。

2. AutoText 域

AutoText 域的语法为：

{ AUTOTEXT *AutoTextEntry* **}**

AutoText 域的指令如表 3-9 所示。

表 3-9 AutoText 域的指令

指令	解释
AutoTextEntry	“自动图文集”词条

AutoText 域的作用是：插入指定的“自动图文集”词条。如果使用 AUTOTEXT 域，而不是直接插入一个“自动图文集”词条，那么在重新定义“自动图文集”词条时，Word 2000 能自动更新文档中该词条的所有实例。

3. Bookmark 和 Ref 域

Bookmark 和 Ref 域的语法为：

{ [**REF**] *Bookmark* [*Switches*] **}**

Bookmark 和 Ref 域的指令和开关如表 3-10 所示。

表 3-10 Bookmark 和 Ref 域的指令和开关

指令或开关	解释
Bookmark	书签名。如果书签所标记的文字包含段落标记，则 BOOKMARK 域之前的文字将使用书签中段落的格式。
\f	增加书签所标记的脚注、尾注或批注序号并插入对应的注释或批注文字。
\h	创建到用书签标记的段落的超级链接。

\n	域将以无后续句点形式显示交叉引用段落的完整的段落编号。不显示上一级别的信息，除非它是当前级的一部分。
\p	使域使用“见上方”或“见下方”显示其相对于源书签的位置。如果文档中REF域在书签之前,则其值为“见下方”。如果文档中 REF 域在书签之后，则其值为“见上方”。如果 REF 域在书签之中，则会返回一条出错信息。
\r	将书签标记段落的无后续句点形式的完整段落编号插入相关文字或相对于编号方案中的位置。
\t	与 \n、\r 或 \w 开关连用时，使 REF 域屏蔽非分隔符或非数字文字。 例如，使用此开关引用“Section 1.01,”时，结果只显示“1.01”。
\w	插入用书签标记的段落的段落编号，此编号会反映该段落在文档全部上下文中的位置。 例如，引用段落“ii.,”时，带 \w 开关的 REF 域可能返回结果为“1.a.ii”。

Bookmark 和 Ref 域的作用是：插入指定的书签所代表的文字或图形。使用时当前文档中必须有该书签的定义。

4. Comments 域

Comments 域的语法为：

{ COMMENTS [*"NewComments"*] **}**

Comments 域的指令如表 3-11 所示。

表 3-11　Comments 域的指令

指令	**解释**
"NewComments"	可选文字，将替代当前“备注”框中的内容。最长 255 个字符，必须放在引号中。

Comments 域的作用是：插入当前文档或模板的“文件”菜单中“属性”对话框“摘要信息”选项卡上的“备注”框中的内容。

例如在域**{** COMMENTS " **{** FILLIN “请更新此次修订的备注信息：” **}** " **}**中，FILLIN 域提示输入新的备注。Word 2000 将用户的响应信息（如：“经理人员审阅时进行的修订”）打印在文档的 COMMENTS 域，并将该响应信息添至“属性”对话框的“备注”框中。

5. Date 域

Date 域的语法为：

{ DATE [\@ "Date-Time Picture"] [*Switches*] **}**

Date 域的开关如表 3-12 所示。

表 3-12 Date 域的开关

开关	作用
\l	插入日期，其格式为最后一次在“插入”菜单中的“日期和时间”对话框中所选的格式。
\@ "Date-Time Picture"	指定替代默认格式的日期格式。如果在“插入”菜单中的“域”命令的“域选项”对话框选中了一种格式，则 Word 2000 将插入相应的日期-时间图片开关。要使用“域选项”对话框未列出的格式，可在“域代码”框键入格式开关。

Date 域的作用是：插入当前日期。单击“页眉和页脚”工具栏上的“插入日期”按钮，可插入一个 DATE 域。除非选择其他格式，否则所显示的日期格式为 Windows“控制面板”上“区域设置属性”对话框的“日期”选项卡或 Windows NT“控制面板”上“国际”对话框中所选的格式。

Date 域的使用示例如表 3-13 所示。

表 3-13　Date 域的示例

域	显示
{ DATE \@ "dddd，MMMM d" }	Saturday，November 26
{ DATE \@ "h:mm am/pm，dddd，MMMM d" }	10:00 am，Saturday，November 26

6. Embed 域

Embed 域的语法为：

{ EMBED *ClassName* [Switches] **}**

Embed 域的指令如表 3-14 所示。

表 3-14　Embed 域的指令和开关

指令或开关	解释
ClassName	容器应用程序名，如 Microsoft Word。不能修改此指令。
\s	在域更新时，将嵌入对象置为原大小。若想保持当前尺寸或对嵌入对象所做的裁剪，则须将此开关删除。
*** mergeformat**	将上次结果所用的尺寸和裁剪应用于新的结果。如果在更新域时要保持上次应用的尺寸和剪切，则不要从域中删除此开关。

Embed 域的作用是：插入支持 OLE 的其他应用程序所创建的对象。使用“插入”菜单中的“对象”命令、“编辑”菜单中的“选择性粘贴”命令或工具栏按钮插入对象（如 Microsoft Excel 工作表对象）时，Word 2000 将插入 EMBED 域。

✍ “域”对话框中没有 EMBED 域，不能人工插入此域。但

可以修改已有的 EMBED 域中的开关。

例如，下面的域将显示文档中嵌入的“Microsoft 图表”对象：

{ EMBED MSGraph * MERGEFORMAT **}**

7. Eq（公式）域

Eq（公式）域的语法为：

{ EQ *Switches* **}**

Eq（公式）域的开关如表 3-15 所示。

表 3-15　Eq（**公式**）**域的开关**

开关	**解释**
数组开关：\a()	绘制一个二维数组
括号：\b()	用括号括住单个元素
位移：\d()	将下一个字符向左或右移动指定磅数
分数：\f(,)	创建分数
分数：\i(,,)	使用指定的符号或默认符号及三个元素创建积分
列表：\l()	将多个值组成一个列表，列表可作为单个元素使用
重叠：\o()	将每个后续元素打印在前一元素之上
根号：\r(,)	使用一个或两个元素绘制根号
上标或下标：\s()	设置上下标
方框：\x()	在元素四周绘制边框

Eq（公式）域的作用是：生成数学公式。开关用于指定如何用括号中的元素建立公式。可用适当的开关选项修改开关。

建议使用“公式编辑器”程序来创建公式。如果没有安装“公式编辑

器”或者想要编写行内公式，可使用 Eq 域。双击 Eq 域时，Word 2000 将此域转换为嵌入的“公式编辑器”对象。注意不能解除 Eq 域的链接。

- 如果要在公式中使用逗号、单括号或反斜杠，可在这些符号前加反斜杠：\,、\(、\\ 。
- 某些开关需要有由逗号或分号隔开的元素列表。如果系统的小数点符号是句号（由 Windows“控制面板”上“区域设置属性”对话框的“数字”选项卡或 Windows NT“国际控制面板”的“数字格式”区指定），此时须用逗号作分隔符。如果系统中的小数点符号是逗号，此时须用分号作为分隔符。

8. Fillin 域

Fillin 域的语法为：

{ FILLIN ["*Prompt*"] [*Switches*] **}**

Fillin 域的指令和开关如表 3-16 所示。

表 3-16　Fillin 域的指令和开关

指令或开关	解释
"Prompt"	显示在对话框中的文字，例如，“请输入客户名：”。
\d "Default"	指定用户没有在提示对话框键入任何信息时的默认响应信息。域{ FILLIN “请输入打字员的姓名缩写：” \d "tds" }在没有输入响应信息时将插入“tds”。如果不指定默认响应，则 Word 2000 使用最后一次输入的响应信息。要将默认值指定为空白，只须在开关后键入空引

	号对，例如,键入“\d ""”。
\o	在邮件合并过程中只提示一次，而不是每次合并新数据记录都提示。在每篇合并结果文档中都插入相同的响应信息。

Fillin 域的作用是：提示用户输入文字。用户的响应信息会打印在域中。每次更新 FILLIN 域时都显示提示。如果 FILLIN 域在邮件合并主文档中，则每次合并新数据记录时显示提示（除非使用了\o 开关）。

当基于包含 FILLIN 域的模板创建新文档时，该域会自动更新。

9. If 域

If 域的语法为：

{ IF *Expression1 Operator Expression2 TrueText FalseText* **}**

If 域的指令如表 3-17 所示。

表 3-17　If 域的指令

指令	**解释**
Expression1 **Expression2**	待比较的值。表达式可以是书签名、字符串、数字、返回一个值的嵌入域或数学公式。如果表达式中有空格，请用引号引住表达式。
Operator	描述
=	等于
<>	不等于
>	大于
<	小于
>=	大于或等于
<=	小于或等于
TrueText **FalseText**	比较结果为真时得到 TrueText，为假时得到的 FalseText。如果没有指定假文字而比较结果为假，

	则 IF 域无结果。每个字符串都必须用引号括住。

If 域的作用是：比较两个值，根据比较结果插入相应的文字。如果用于邮件合并主文档，则 IF 域可以检查合并数据记录中的信息，如邮政编码或帐号等。例如，可只发信给某市的客户。

✍ 如果操作符是=或<>，则 *Expression2* 可用问号（?）表示任意单个字符，用星号（*）表示任意字符串。表达式必须在引号内才能作为字符串比较。如果 *Expression2* 中使用了星号，则 *Expression1* 中对应于星号的部分加上 *Expression2* 中的其余字符，总数不能超过 128 个字符。

10. IncludePicture 域

IncludePicture 域的语法为：

{ INCLUDEPICTURE "*FileName*" [*Switches*] **}**

IncludePicture 域的指令如表 3-18 所示。

表 3-18　IncludePicture 域的指令

指令	解释
"FileName"	图形文件名称和位置。如果其中包含较长的带空格文件名，请用引号引住。指定路径时，请以双反斜杠替代单反斜杠。例如："C:\\Manual\\Art\\Art 22.bmp"
\c Converter	标识所需的图形过滤器。图形过滤器名中不需要文件扩展名 .flt。例如，想使用图形过滤器 Pictim32.flt，只需键入："pictim32"。
\d	图形数据不随文档保存以减小文件长度。

IncludePicture 域的作用是：插入指定的图形。插入INCLUDEPICTURE 域时，将指针指向“插入”菜单中的“图片”子菜单，单击“来自文件”命令，再单击“插入”按钮旁的箭头，然后选中“链接文件”复选框。

✍ 可用 INCLUDEPICTURE 域替代以前 Word 版本中使用的 IMPORT 域。如果打开包含 IMPORT 域的文档，这些域仍将保留在文档中且依然有效。

如果双击 INCLUDEPICTURE 域所插入的图形，Word 会显示“设置图片格式”对话框。如果不使用绘图工具更改图形，可在创建图形的应用程序中编辑图形，然后在 Word 中更新此域。

如果 Word 2000 不能识别图形文件格式，可检查“插入图片”对话框的“文件类型”框（指针指向“插入”菜单中的“图片”子菜单，然后单击“来自文件”命令）。框中列出系统中安装的图形过滤器。

11. IncludeText 域

IncludeText 域的语法为：

{ INCLUDETEXT "*FileName*" [*Bookmark*] [*Switches*] **}**

IncludeText 域的指令和开关如表 3-19 所示。

表 3-19　IncludeText 域的指令和开关

指令或开关	解释
"FileName"	文档名称和位置。如果其中包含较长的带空格文件名，

	用引号引住。指定路径时，以双反斜杠替代单反斜杠。例如："C:\\My Documents\\Manual.doc"
Bookmark	书签名，引用文档中要包含的部分。
\!	禁止 Word 更新插入文字中的域，除非此域先在源文档中得到更新。

IncludeText 域的作用是：插入命名文档中包含的文字和图形。可插入整篇文档；如果是 Word 文档，可只插入由书签引用的部分。

✍ 如果源文档是 Word 文档，可在 INCLUDETEXT 域中编辑插入的文字结果并将更改存回源文档。方法是：编辑插入的文字，然后按 Ctrl+Shift+F7。

例如用以下域插入文档中由书签"Summary"所引用的部分：

{ INCLUDETEXT "C:\\Winword\\Port Development RFP" Summary **}**

12. Index 域

Index 域的语法为：

{ INDEX [*Switches*] **}**

Index 域的开关如表 3-20 所示。

表 3-20　Index 域的开关

开关	说明
\b 书签	为文档中由指定书签标记的部分建立索引。例如，域 { INDEX \b Select } 为文档中由书签"Select"标记的部分建立索引。
\c 列	在一页上建立多于一栏的索引。例如，域 { index \c 2 } 可建立一个两栏的索引。最多可以指定四栏索引。

\d "分隔符"	与 \s 开关连用时，指定序列号与页码之间的分隔符（最多为五个字符）。例如，域{ INDEX \s chapter \d " : " } 以“2 : 14”的形式显示页码。如果省略了 \d 开关，就用连字符 (-) 作为分隔符。分隔符需用引号引起来。
\e "分隔符"	指定索引项和页码之间的分隔符（最多为五个字符）。例如，域 { INDEX \e "; " } 在索引中给出这样的结果：“Inserting text; 3”。如果省略了 \e 开关，那么用一个逗号和一个空格 (,)作分隔符。分隔符需用引号括起来。

续　表

开关	说明
\f "标识符"	只用指定类型的索引项建立索引。例如，域{ INDEX \f "a" } 所生成的索引中只包含用象{ XE "Selecting Text" \f "a" } 这样的 XE 域标明的项目。默认的项目类型是“I”。
\g "分隔符"	指定表示页面范围时所用的分隔符(最多为五个字符)。这些分隔符必须用引号引起来。默认的分隔符是一条短划线 (-)。例如，域 { INDEX \g " to " } 会显示出这样的页面范围：“Finding text, 3 to 4”。
\h "标题"	在索引中按字母顺序排列的各组索引项之间插入具有“索引标题”样式的文本。文本必须用引号括起来。例如，域{ INDEX \h "– A– "} 在索引中按字母顺序排列的各组索引项之前显示该组对应的字母。要在各组之间插入一空白行，可用空引号：\h ""。
\l "分隔符"	指定多页引用间的分隔符。默认分隔符是一个逗号和一个空格 (,)。可以使用多达五个字符，但这些分隔符必须用引号引起来。例如，域{ INDEX \l " or " } 在索引中给出这样的结果“Inserting text, 23 or 45 or 66”。
\p "范围"	根据指定的字母生成索引。例如，域 { INDEX \p a-m } 生成一个只含字母 A 到 M 的索引。要在索引中包括以非字母开头的项，可使用感叹号 (!)。例如，由域 { INDEX \p !--t } 生成的索引含有任意特殊字符以及字母 A 到 T。
\r	把次索引项与主索引项放在同一行中。主索引项与次索

	引项之间用冒号 (:) 分隔；次索引项之间则用分号 (;) 分隔。域 { INDEX \r } 给出这样的结果："Text: inserting 5, 9; selecting 2; deleting 15"。
\s	其后跟有序列名时，将序列号添加到页码中。序列号与页码间的默认分隔符为连字符 (-)，可用 \d 开关来指定其他的分隔符。

Index 域的作用是：建立并插入一个索引。INDEX 域收集由 XE（索引项）域指明的索引项。INDEX 域可用"插入"菜单中的"索引和目录"命令插入。

例如，域**{** INDEX \s chapter \d "." **}**为主控文档建立索引。每个子文档是一章，章节标题中包括一个给章节编号的 SEQ 域。\d 开关用句号（.）作为序列号与页码之间的分隔符。对某 Word 文档用该域产生的索引如下所示：

亚里士多德, 1.2

地球, 2.6

木星, 2.7

火星, 2.6

13. MacroButton 域

MacroButton 域的语法为：

{ MACROBUTTON *MacroName DisplayText* **}**

MacroButton 域的指令如表 3-21 所示。

表 3-21　MacroButton 域的指令

指令	解释
MacroName	双击域结果时运行的宏名。活动文档模板或通用模板中必须有要运行的宏。
DisplayText	显示为“按钮”的文字或图形。可使用结果为文字或图形的域，如 BOOKMARK 或 INCLUDEPICTURE。在域结果中，文字或图形必须在一行内，否则会出错。

MacroButton 域的作用是：插入宏命令，双击 MACROBUTTON 域结果就可运行该宏。也可单击 MACROBUTTON 域，然后按 Alt+Shift+F9 组合键。

14. Page 域

Page 域的语法为：

{ PAGE [* *Format Switch*] **}**

Page 域的开关如表 3-22 所示。

表 3-22　Page 域的开关

开关	说明
*** Format Switch**	可选开关，该开关可替代在“页码格式”对话框（单击“插入”菜单中的“页码”命令可显示该对话框）的“数字格式”框中选择的数字样式。 要改变页码的字符格式，可修改“数字格式”框中字符样式。

Page 域的作用是：在 PAGE 域所在处插入页码。单击“插入”菜单中的“页码”命令或单击“页眉和页脚”工具栏上的“插入页码”按钮可以插入 PAGE 域。

15. Quote 域

Quote 域的语法为：

{ QUOTE "*LiteralText*" **}**

Quote 域的指令如表 3-23 所示。

表 3-23 Quote 域的指令

指令	说明
"LiteralText"	插入的文字。这些文字必须用引号括起来。并可包含除 AUTONUM、AUTONUMLGL、AUTONUMOUT 或 SYMBOL 外的其他任何域。

Quote 域的作用是：将指定文字插入文档。

例如，下面嵌套有 IF、=(FORMULAR)和 DATE 域的 QUOTE 域可以产生上个月的名称。如果现在是二月，“一月”就是域的结果。该示例可用于事后的报告（如，销售报告）。

{ QUOTE **{** IF **{** DATE \@ "M" **}** = 1 "12" " **{**= **{** DATE \@ "M" **}** -1 **}** " **}**/1/95 \@ "MMMM" **}**

16. Set 域

Set 域的语法为：

{ SET *Bookmark* "*Text*" **}**

Set 域的指令如表 3-24 所示。

表 3-24 Set 域的指令

指令	说明

Bookmark	要用来代表信息的书签名。例如，InterestRate。
"Text"	书签所代表的信息。文本需用引号括起来，数字不必用引号括起来。此信息可为嵌套域的结果。

Set 域的作用是：定义指定书签名所代表的信息。可以在宏中引用书签，或在其他域中包含该书签，如 IF 域。若要打印该信息，则必须在文档中插入一个 REF 域或 BOOKMARK 域。

例如在下例中，如果在 FILLIN 发出提示时输入 3 作为应答信息，那么书签 TotalCost 的结果值是 $82.50。BOOKMARK 域用来打印价格和订货者姓名。数字图片开关用来将结果显示为带货币符号的值。注意第一个 SET 域中用了引号。

域:

{ SET EnteredBy "Maria Gerard" **}**

{ SET UnitCost 25.00 **}**

{ SET Quantity **{** FILLIN "Enter number of items ordered:" **} }**

{ SET SalesTax 10% **}**

结果：

{ SET TotalCost **{** = (UnitCost * Quantity) + ((UnitCost * Quantity) * SalesTax) **} }**

Total cost: **{** TotalCost \# "$#0.00" **}**

Thank you for your order,

{ EnteredBy **}**

17. Time 域

Time 域的语法为：

{ TIME [\@ "Date-Time Picture"] **}**

Time 域的指令如表 3-25 所示。

表 3-25　Time 域的指令

指令	说明
\@ "Date-Time Picture"	指定一种不同于默认设置的时间格式。如果在“域选项”对话框（单击“插入”菜单中的“域”命令可显示该对话框）中选择了一种格式，那么 Word 将在域中插入相应的日期-时间图片开关。 要使用“域选项”对话框中没有的格式，可在域代码中键入格式开关。

Time 域的作用是：插入当前时间。单击“页眉和页脚”工具栏上的“插入时间”按钮，即可插入一个 TIME 域。除非指定了日期-时间图片开关，否则时间以“时间”选项卡（该选项卡位于 Windows“控制面板”的“区域设置属性”对话框中）中设置的格式显示。

✍ 如果使用的是“插入”菜单中的“日期和时间”命令，并选中了“自动更新”复选框，则 Word 将插入一个 TIME 域。根据所选择的格式，Word 2000 用日期-时间图片开关来显示当前日期或时间或同时显示二者。例如，如果在“日期与时间”对话框中选择了“10/12/95 10:11AM”，那么

Word 将插入域**{** TIME \@ "MM/dd/yy h:mm AM/PM" **}**。

Time 域的使用示例如表 3-26 所示。

表 3-26　Time 域的示例

域	结果
{ TIME }	11:11 AM（或在 Windows“控制面板”的“区域设置属性”对话框中指定的默认时间格式）
{ TIME \@ "h am/pm" }	11am

3.2　宏

有时候，用户需要在 Word 2000 中作大量重复性的工作。这时候，使用 Word 中提供的宏可能是最好的方法。

3.2.1　宏的概述

宏是建立在 Microsoft Visual Basic Application（即 VBA，在后面将进行详尽的介绍）之上的。它是一系列组合在一起的 Word 命令和指令，它们形成了一个命令，以实现任务执行的自动化。可以将宏看作一个由一串 Word 命令和操作构成的小型程序。可以创建并执行宏（宏实际上就是一条自定义的命令），以替代人工进行的一系列费时而单调的重复性 Word 操作，自动完成所需任务。

宏的一些典型应用包括：

（1）加速日常编辑和格式设置。

（2）组合多个命令，例如插入具有指定大小、边框、行数和列数的表格。

（3）使对话框中的选项更易于访问。

（4）自动执行一系列复杂的任务。

3.2.2　宏的创建

宏的创建有两种方法：使用宏录制器录制一个宏或者在 Visual Basic 编辑器中编辑一个宏。前者使用起来非常方便，但是它有一定的局限性（比如某些操作不能被录制），只能用于创建比较简单的宏；后者的功能比较强大，可以创建比较复杂的宏，缺点是比较费时费力且需要了解较多的关于 VBA 的知识。

1. 通过宏录制器来录制一个宏

先来看一看怎样通过宏录制器来录制一个宏。

录制一个宏的步骤如下：

（1）打开“工具”菜单。

（2）选择“宏”子菜单中的“录制新宏”选项，此时将出现如图 3-7 所示的“录制宏”对话框。



图 3-7 “录制宏”对话框

（3）在“宏名”框中输入宏的名字。

（4）在“将宏保存在”下拉框中选择宏保存的位置。

（5）可以选择将宏保存在“所有文档（Normal.dot）”中，这样在任何一个打开的文档中都可以使用创建的宏。如果不想在其他的文档中使用这个宏，可以选择将宏保存在当前文档中。

（6）在“说明”框中，键入对宏的说明。

（7）单击“确定”按钮，即可开始宏的录制。此时“录制宏”对话框被关闭，出现如图 3-8 所示的“停止录制”对话框。



图 3-8 “停止录制”对话框

（8）现在就可以正式开始宏的录制了。宏录制器会忠实地记录用

户的所有操作。如果不想把某些操作记录到宏中，可以先点击“暂停录制”按钮，等到执行完这些操作以后，再点击“恢复录制”按钮继续录制宏。

✍ 在录制宏时，可用鼠标单击命令和选项。但是，宏录制器不能录制鼠标在文档窗口中的运动。要录制如移动插入点，或者选定、复制及移动文本这样的操作，必须使用快捷键。

（9）在执行完所有需要录制的操作以后，单击“停止录制”按钮，即可结束宏的录制。

在图 3-7 所示的“录制宏”对话框中，有两个图形按钮“工具栏”和“键盘”。单击“工具栏”可以将要录制的宏指定到工具栏或者菜单中；单击“键盘”按钮可以为宏指定一个快捷键。如果用户经常要使用到某一个宏的话，这样的设置是十分方便的。

为宏指定快捷键的步骤如下：

（1）按照前面的方法打开图 3-7 所示的“录制宏”对话框。

（2）单击“录制宏”对话框中的“键盘”按钮，此时将出现如图 3-9 所示的“自定义键盘”对话框。

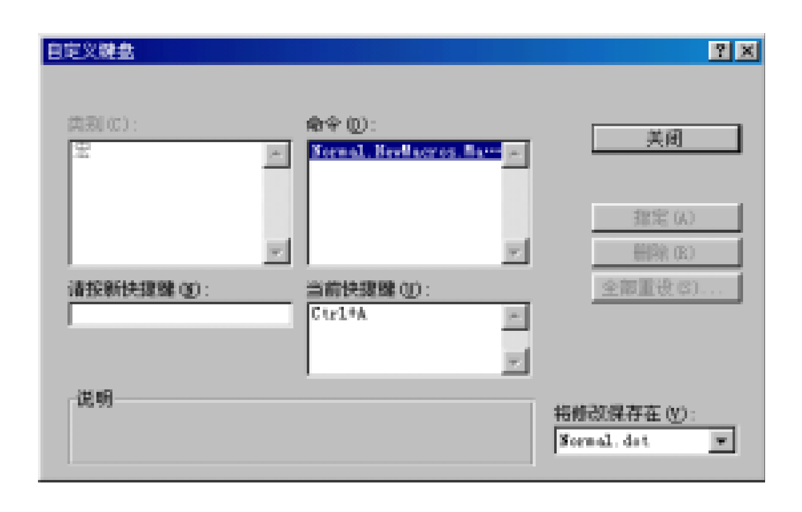


图 3-9　“自定义键盘”对话框

（3）按下想要为宏指定的快捷键（例如“Ctrl＋A”），此时组合键将出现在“请按新快捷键”框中。

（4）单击“指定”按钮，将快捷键指定给宏。如果指定成功的话，组合键将出现在“当前快捷键”框中，如图 3-9 所示。

（5）单击“关闭”按钮，开始宏的录制。

也可以将宏指定到工具栏或菜单中，操作步骤如下：

（1）按照前面的方法打开图 3-7 所示的“录制宏”对话框。

（2）单击“工具栏”对话框，此时将出现如图 3-10 所示的“自定义”对话框。

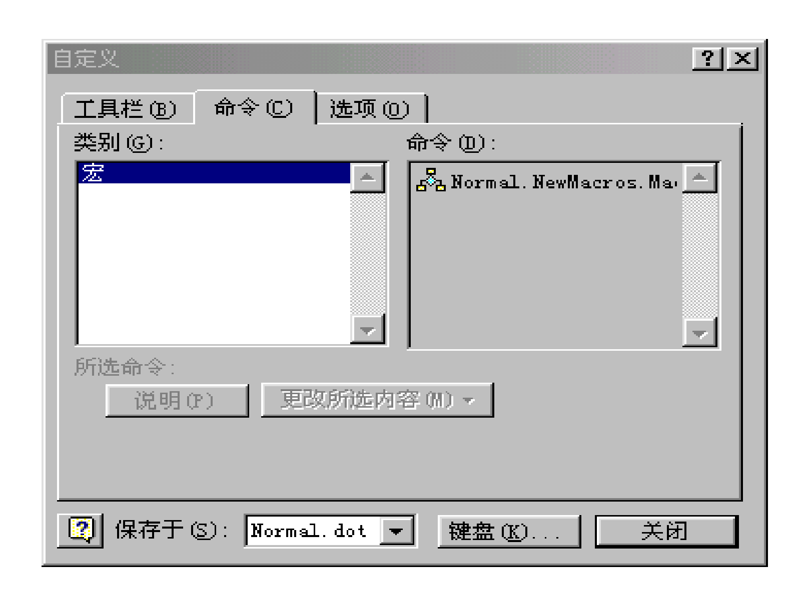


图 3-10　“自定义”对话框

（3）从“命令”框中选中一个宏，将它拖到工具栏或菜单的合适位置即可。

（4）由于新建宏的名字通常很长（例如Normal.NewMacros.Macro1），这样在工具栏上很占地方。如果想使工具栏更加简洁、美观，可以单击工具栏上新建宏的快捷按钮，在快捷按钮的周围出现黑色边框以后，点击“自定义”对话框中的“更改所选内容”按钮，此时将出现如图 3-11 所示的快捷菜单。

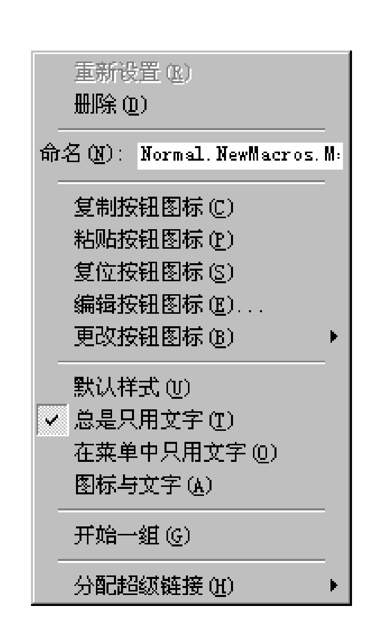


图 3-11　“更改所选内容”快捷菜单

（5）从快捷菜单中可以看到，快捷按钮在工具栏中的缺省显示方式是“总是只用文字”，将其该为“缺省”方式后，快捷按钮在工具栏中将以图标的方式显示，如图 3-12 所示。也可以通过快捷菜单中的“更改按钮图标”选项来更改快捷按钮的图标。



图 3-12　修改前和修改后的快捷按钮

（6）单击“键盘”按钮可以为宏指定一个快捷键，方法同前。

（7）单击“确定”按钮，即可开始宏的录制。

2. 用 Visual Basic 编辑器创建宏

在前面已经提到，Word 2000 中的宏是建立在 Microsoft Visual Basic

Application 之上的。可以使用 Visual Basic 编辑器来创建功能强大的宏，它能够创建用“宏录制器”不能录制的功能。也可以用 Visual Basic 编辑器对宏进行编辑和调试。

使用 Visual Basic 编辑器创建宏的方法如下：

（1）打开“工具”菜单并选择其中的“宏”命令打开“宏”子菜单。

（2）选择子菜单中的“宏”命令，此时将出现如图 3-13 所示的“宏”对话框。

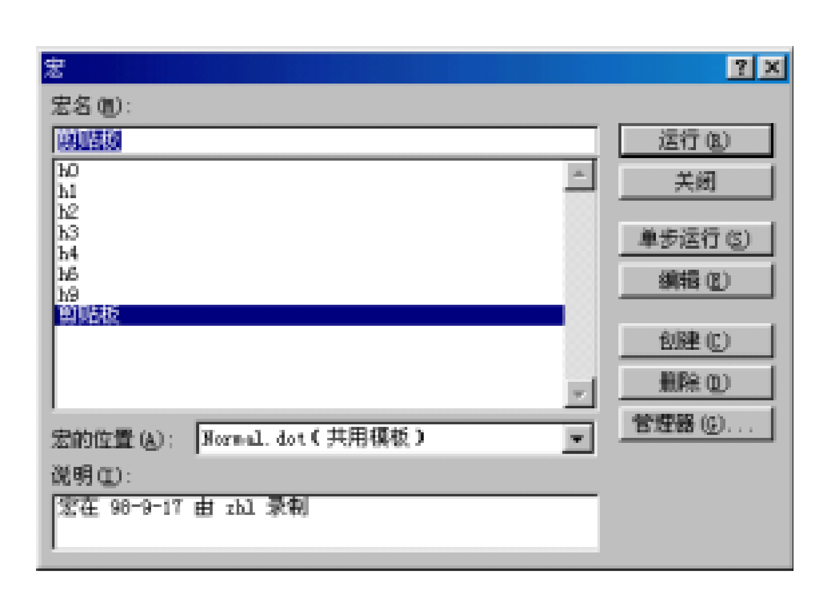


图 3-13　“宏”对话框

（3）在“宏名”框中输入要创建的宏的名字。

（4）单击“确定”按钮，即可打开 Visual Basic 编辑器。使用 Visual Basic 编辑器编辑宏的具体内容将在本书后面的相关章节中介绍。

3.2.3　宏的运行

创建完一个宏以后，就可以在文档中运行它了。如果把宏指定到了工具栏、菜单或者给宏指定了快捷键，可以使用指定的快捷方式来运行这个宏。如果没有给宏指定快捷方式，那么可以从图 3-13 所示的“宏”对话框中运行一个宏，具体步骤如下：

（1）打开“工具”菜单中的“宏”子菜单。

（2）选择“宏”命令，打开如图 3-13 所示的“宏”对话框。

（3）在列表框中选择要运行的宏。如果该宏没有出现在列表中，可从“宏的位置”框中选择其他文档、模板或列表。

（4）单击“运行”按钮后，Word 2000 将在当前打开的文档中运行该宏。

（5）如果想要检验一下宏的每一步的运行效果，可以单击“单步运行”按钮来单步运行一个宏。每单击一次“单步运行”按钮，宏就执行一个操作，直至到达宏的末尾处为止。

（6）在宏的运行过程中，可以将其随时终止。终止一个宏的运行的方法为：在宏运行时按下“Esc”键，此时 Word 2000 将显示一个信息表明宏将被终止，然后单击“确定”按钮即可。

3.2.4　自动宏

在 Word 2000 中存在一些特殊的宏，它们会在某些特定的时机（如

启动 Word 2000 时，打开一个文档等，关闭一个文档时等等）自动运行。这样的宏被称为自动宏。Word 2000 中共存在如表 3-27 所示的五个自动宏。

表 3-27　Word 2000 中的自动宏

自动宏	功能
AutoExec	在启动 Word 2000 时自动运行
AutoNew	在新建一个文档时自动运行
AutoOpen	在打开一个文档时自动运行
AutoClose	在关闭一个文档时自动运行
AutoExit	在退出 Word 2000 时自动运行

自动宏的创建方法和其他宏的创建方法是一样的。在创建自动宏时，可以选择将宏保存在“通用文档模板”（Normal.dot）或者用户自己创建的模板中。保存在通用文档模板中的宏称为全局宏，保存在用户自建立的模板中的宏称为局部宏。如果自动宏是一个全局宏的话，用户每次打开、新建或关闭任何文档时自动宏都将被运行；如果自动宏是一个局部宏的话，只有用户基于此模板新建一个文档，或者打开/关闭基于此模板的文档时，自动宏才会被运行。显而易见，Autoexec 和 Autoexit 自动宏只有在是全局宏时，才会起作用。

✍ 如果用户不想在某个特定的时机运行自动宏，可以按下“Shift”键。例如不想在启动 Word 2000 时运行 Autoexec 自动宏，只需在启动 Word 2000 时按下“Shift”键，即可跳过自动宏的运行。

3.2.5　宏的管理

下面将介绍如何管理模板中的宏。宏的管理主要包括以下三个方面的内容：宏的复制，宏的删除，以及重命名宏。

1．宏的复制

有时，用户会发现某个录制的全局性的宏，只是在一两个模板或文件中才用得到；有时，用户想使某个特殊模板中录制的宏在所有的文档中都能应用；有时又希望将别人文档中的宏复制到自己的模板或文档上。为了做到这一点，Word 2000 提供了在模板之间对宏进行复制和移动的服务。

将宏从一个模板或文档中复制到另一个文档或模板中的方法为：

（1）单击“工具”菜单中“宏”子菜单中的“宏”命令，打开图 3-13 所示的“宏”对话框。

（2）单击“宏”对话框中的“管理器”按钮，此时将出现如图 3-14 所示的“管理器”对话框。

（3）在“管理器”对话框中，左边的列表是当前文件包含的宏，而右边的列表是通用文件模板（Normal.dot）所包含的宏。可以在一个列表中选择一个宏方案项，然后单击“复制”按钮，将其复制到另外一个列表中去。

（4）如果想在任意的两个文件或模板之间进行宏的复制，则可以先单击两边的“关闭文件”按钮将原来的两个文件关闭，然后单击两

边的“打开文件”按钮，打开想要进行宏的复制的文件或模板，此时就可以按照上面描述的方法进行宏的复制。

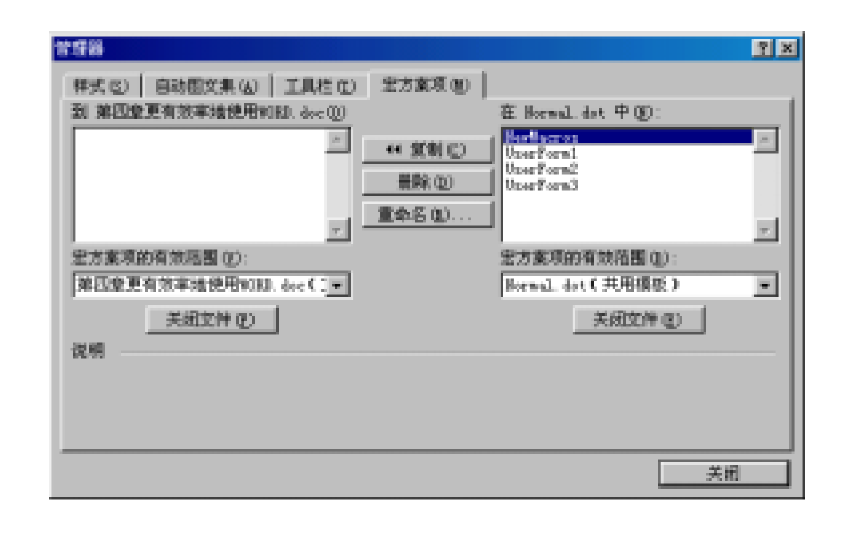


图 3-14　“管理器”对话框

✍ 用以上方法进行的是宏方案的复制，即把一个文件或模板中的所有宏复制到另一个文件或模板。如果要复制单个的宏，可单击“工具”菜单中“宏”子菜单的“宏”命令，在“宏”对话框中选定要复制宏后，单击“编辑”按钮，然后用“Visual Basic 编辑器”的标准编辑功能完成复制操作。

2. 宏的删除

当某个宏已不再需要时，可以将其删除，以加快程序运行的速度。删除宏的方法如下：

（1）单击“工具”菜单中“宏”子菜单中的“宏”命令，打开图 3-13 所

示的“宏”对话框。

（2）在“宏名”框中单击要删除的宏的名称。如果该宏没有出现在列表中，可在“宏的位置”框中选择其他的宏列表。

✍ 要删除多个宏，可在按下“Ctrl”键的同时单击“宏名”框中要删除的各个宏，然后单击“删除”按钮。

用户也可以选择删除宏方案，即将文档中的一组宏的集合同时删除。

删除宏方案的方法如下：

（1）单击“工具”菜单中“宏”子菜单中的“宏”命令，打开图 3-13 所示的“宏”对话框。

（2）单击“管理器”按钮打开图 3-14 所示的“管理器”对话框。

（3）在“管理器”对话框中，由两个列表中的一个选择要删除的宏方案，然后单击“删除”按钮。

（4）Word 2000 将询问是否真正要删除宏，单击按钮“是”。

（5）关闭管理器和宏对话框。

3. 重命名宏

在 Word 2000 中，既可以给文档中的宏方案重命名，也可以给单个宏重命名。

给宏方案重命名是通过“管理器”进行的，具体步骤为：

（1）单击“工具”菜单中的“宏”子菜单的“宏”命令，打开图 3-13 所

示的“宏”对话框。

（2）单击“管理器”按钮打开图 3-14 所示的“管理器”对话框。

（3）在“管理器”对话框的任意一个列表中，选择需要重命名的宏方案，然后单击“重命名”按钮。此时将出现如图 3-15 所示的“重命名”对话框。



图 3-15　“重命名”对话框

（4）在“新名称”框中键入宏方案的新名称后，单击“确定”按钮。

（5）关闭管理器和宏对话框。

✍ 给单个宏命名需要通过 Visual Basic 编辑器，具体内容将在本书后面的相关章节中介绍。

3.2.6　宏的安全性

宏的使用给处理文档带来了巨大的方便，然而事物总是具有两面性的，既有有利的一面，也必然有其不利的一面。那么，宏的另一面是什么呢？答案之一就是令人望而生畏的“宏病毒”。

病毒是一种计算机病毒，它保存在文档、模板或加载项中的宏中。如果打开了这样的文档或执行了触发宏病毒的操作，就可能激活宏病

毒，将其传播到计算机中，并保存在“Normal”模板或其他共用模板中。此后打开的每个文档都会自动“感染”上宏病毒，如果其他人打开了这些被感染的文档，宏病毒将会传播到他们的计算机中。

由此可见，宏病毒的破坏力是相当惊人的。尽管如今有很多的杀毒软件，但是预防宏病毒的最好方法还是不打开带有宏病毒的文档或模板。为此 Word 2000 给宏设定了安全级。当安全级别比较高时，Word 2000 将不打开来历不明的宏。

设置宏的安全级别的方法为：

（1）打开“工具”菜单中的“宏”子菜单。

（2）选择“安全性”命令，将出现如图 3-16 所示的“安全性”对话框。

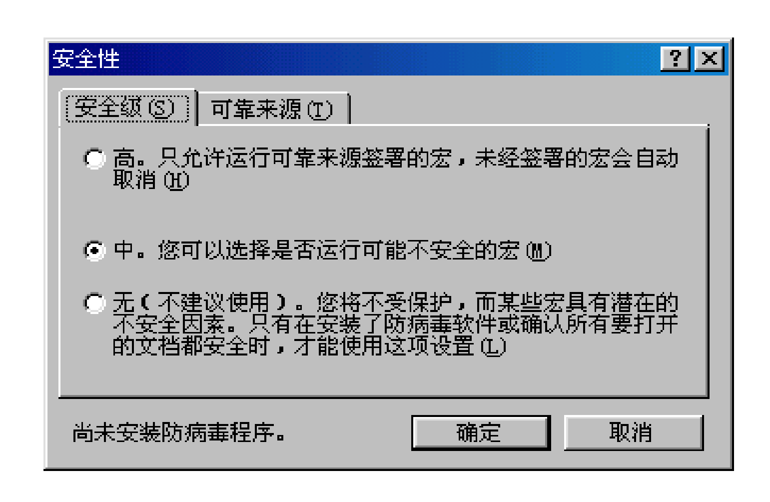


图 3-16　宏的“安全性”对话框

（3）为文档中的宏选择安全级别。安全级别共有三种：“高”，“中”，“低”。

（4）单击“确定”按钮关闭对话框。

Word 2000 中宏的三种安全级别的功能如下：

（1）高：只能运行可靠来源签署的宏，未经签署的宏在文档中被禁用。在安全级别设置为高时，一般不需要安装防“宏病毒”的软件。

（2）中：可以运行有可靠来源签署的宏，如果 Word 2000 遇到的宏的来源不在可靠来源列表中，则会显示警告信息。让用户选择在打开文档时启用还是禁用宏。

（3）无：如果用户确信所有要打开的文档和加载项都是安全的，可以选择此选项，它将关闭 Word 2000 的宏病毒防护功能。在此安全级下，打开文档时将自动启用宏。

3.3　拼写和语法检查

在键入文档时，出现错误总是难免的。而更正错误的工作更是让人头痛。好在 Word 2000 提供了几种效率工具来帮助完成文档的最后校订。这几种效率工具是：拼写检查，语法检查，同义词库和自动更正功能。在本节和下一节中，将分别介绍 Word 2000 的拼写和语法检查功能以及自动功能。

3.3.1　拼写和语法检查的操作

当 Word 认为文档中的某处地方出现了拼写和语法错误时，它将在

出现错误的地方加上红色或者绿色的下划波浪线。

尤其令人振奋的是，Word 2000 除了可以对英文进行拼写和语法的检查之外，还可以对简体中文进行校对。中文校对的用户界面跟英文校对的界面几乎完全相同，只有一点例外：在此版本中，没有中文用户词典可用来“添加”中文词语，因此用户不能添加诸如姓名这样的自定义词语，这一点将会在后面的示例中了解到。此外，中文校对只能处理简体中文字符。如果使用的是繁体中文输入法或将简体中文转换成了繁体中文，则中文校对功能将不会处理这些文字。

可以通过前台或后台两种方法对出现拼写和语法错误的语句进行校对。

1. 前台校对

对文档进行前台校对的具体步骤为：

（1）选中想要进行拼写和语法校对的文本。如果不选择任何文本，则将对全文进行校对。

（2）单击“常用”工具栏上的“拼写和语法”按钮，或者单击“工具”菜单中的“拼写和语法”命令，又或者按下快捷键“F7”，均可打开如图 3-17 所示的“拼写和语法”对话框。

（3）“拼写和语法”对话框中有两个文本框，上方的“不在词典中”文本框显示的是 Word 2000 认为出错的语句，下方的文本框显示的则是 Word 2000 的更正建议。对话框中几个按钮的作用如下：

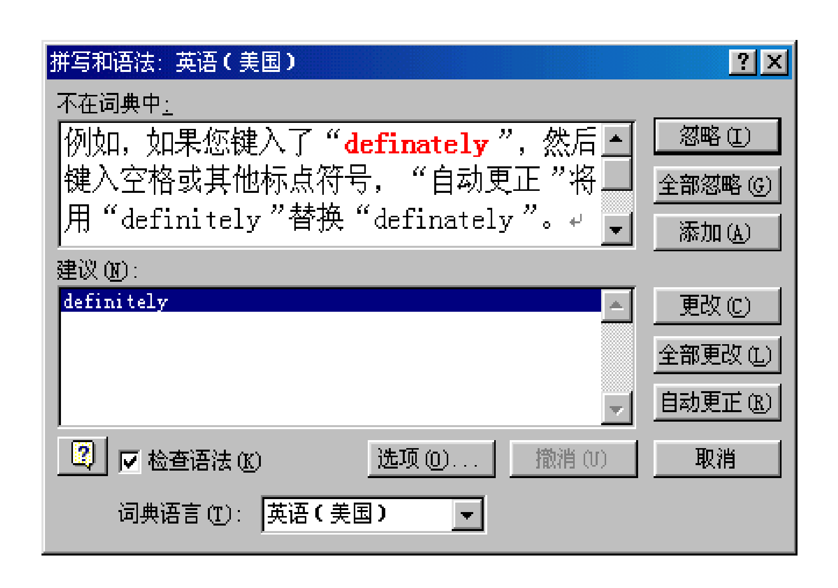


图 3-17　“拼写和语法”对话框

- “忽略”按钮：　单击此按钮将不更改突出显示的错误部分并查找下一个拼写和语法错误。
- “全部忽略”按钮：　单击此按钮将不更改所有突出显示的错误部分并查找下一个拼写和语法错误。在当前的 Word 操作中，将忽略这项拼写错误或者这种类型的语法错误。
- “添加”按钮：　将“不再词典中”文本框中突出显示的文本添加到用户的自定义词典之中。此后 Word 2000 将不再认为此种拼写是错误的。
- “更改”按钮：　单击此按钮将会把错误的语句更改为“建议”文本框中选择的内容。也可以先编辑“建议”文本框中的内容，再单击“更改”按钮进行替换。
- “全部更改”按钮：　单击此按钮将把文档中所有出现在“不

在词典中”的语句更改为“建议”文本框中选择的内容。也可以先编辑“建议”文本框中的内容，再单击“更改”按钮进行替换。

- “自动更正”按钮：　单击此按钮将把错误的语句及其更正方案添加到“自动更正“列表中，以便以后在键入时自动更正。

因此，如果用户想要更改 Word 2000 给出的错误语句，可以单击“更改”、“全部更改”、“自动更正”按钮；如果不想更改 Word 2000 给出的错误语句，则可以单击“忽略”、“全部忽略”和“添加”按钮；如果用户想要撤销最近所做的拼写和语法检查操作，可以单击“撤销”按钮。

2. 后台校正

Word 2000 还可以进行后台的拼写和语法校正。在进行后台校正时，在 Word 2000 认为出现语法和拼写错误的语句下面，将显示红色或绿色的波形下划线。用鼠标右键单击出现错误的语句，弹出的快捷菜单如图 3-18 和 3-19 所示。

oddest　　　　　　宏对话框



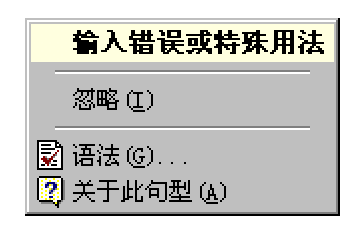


图 3-18　英文校正的快捷菜单　　图 3-19　中文校正的快捷菜单

图 3-18 和图 3-19 所示的分别是英文校对和中文校对的快捷菜单。可以看到尽管 Word 2000 支持中文的校对，但是其功能还是不完善的。它没有中文用户词典可用来“添加”中文词语。希望 Word 2000 的下一个版本中能够完善中文校对的功能。

下面针对英文校对的快捷菜单来介绍 Word 2000 的后台校对。

（1）修改拼写和语法错误：快捷菜单的最上方的单词列表是 Word 2000 提供的更正建议，如果选择了其中的一个单词，文档中错误的单词或语句就会被所选择的正确的单词代替。

（2）忽略错误：如果用户认为输入的单词是正确的，只是由于 Word 2000 不能识别而给其标上了错误标记，则可选择快捷菜单中的“全部忽略”命令。在当前的 Word 2000 操作中，再次输入该单词或语句时，Word 2000 将不再将其标注为错误。

（3）将单词添加到单词列表中：如果用户不仅希望 Word 2000 忽略对该单词的检查，而且还希望将这个单词加入 Word 2000 的词库，使它成为 Word 2000 能够识别的单词，可以选择快捷菜单中的“添加”命令。

3.3.2　语法检查的设置

有时，Word 2000 提供的缺省的拼写和语法检查的设置并不符合要求。可以更改这些设置，更改的方法如下：

（1）选择“工具”菜单栏中的“拼写和语法”选项，或者按下快捷键“F7”，打开图 3-17 所示的“拼写和语法”对话框。

（2）单击“选项”按钮，将出现如图 3-20 所示的“拼写和语法”选项对话框。

（3）在“拼写和语法”选项对话框中，根据需要对拼写和语法的各选项进行设置。



图 3-20 “拼写和语法”选项对话框

3.3.3 显示或隐藏自动拼写或语法检查所用的波形下划线

（1）单击“工具”菜单中的“选项”命令，然后单击“拼写和语法”选项卡。

（2）确保已选中“键入时检查拼写”和（或）“键入时检查语法”复选框。

（3）执行下列一项或多项操作：

- 欲显示或隐藏标记错误拼写的下划线，应选中或清除“隐藏文档中的拼写错误”复选框。
- 欲显示或隐藏标记错误语法的下划线，应选中或清除“隐藏文档

中的语法错误”复选框。

3.3.4　进行拼写和语法检查时跳过文本

（1）选定不必检查的文本。

（2）单击“工具”菜单中“语言”子菜单中的“设置语言”命令。

（3）选中“不检查拼写或语法”复选框。

3.3.5　打开或关闭自动拼写和语法检查功能

（1）单击“工具”菜单中的“选项”命令，然后单击“拼写和语法”选项卡。

（2）执行下列一项或多项操作：

- 欲打开或关闭自动拼写检查功能，应选中或清除“键入时检查拼写”复选框。
- 欲打开或关闭自动语法检查功能，应选中或清除“键入时检查语法”复选框。

✍ 打开自动拼写和语法检查功能时，Word 2000 将清除“隐藏文档中的拼写错误”和“隐藏文档中的语法错误”复选框。这样可使拼写和语法检查用波形下划线标记可能的错误。如果波形下划线分散注意力，可以选中这些复选框，隐藏波形下划线。

3.4　自动功能

Word 2000 的自动功能就是指在文档的编辑过程中，Word 2000 自动执行的格式和命令。Word 2000 的自动功能大大方便了文档的编辑，使文字处理的办公自动化程度得到了进一步的提高。

Word 2000 的主要自动功能包括：

（1）自动更正功能。

（2）自动套用格式。

（3）自动图文集。

（4）自动为文档编写摘要。

以下将分别介绍各种自动功能。

3.4.1　自动更正

键入文档时，难免会出现这样或那样的错误，如果要对每一个错误进行修改的话，显然会大大地影响编辑文档的速度。使用 Word 2000 的“自动更正”功能可以在键入时自动更正许多常见的键入、拼写和语法错误。例如键入“做威做福”，Word 2000 将会自动将其更正为“作威作福”。

此外，使用自动功能可以快速输入一些经常使用但又不愿一次次地重新键入或插入的文字、符号和图形，或者一些特殊的符号。例如可以输入 ms，然后将其自动更正为 Microsoft。又如要想输入右箭头

“→”（键盘上没有这个符号），可以输入“- ->”，Word 2000将自动将其更正为右箭头符号。

打开“工具”菜单并选择其中的“自动更正”命令，将出现如图3-21所示的“自动更正”对话框。

可以通过选择或清除“自动更正”对话框中的复选框来定制“自动更正”的功能。例如，想要使文档中的日期是小写的，则必须先清除“日期大写”复选框。

有关“自动更正”功能的一些高级操作如下：

1. 编辑“自动更正”词条

“自动更正”是通过使用名为“自动更正”词条的内置更正项列表，来检测并更正键入错误、误拼的单词、语法错误和常用符号的。我们可以通过编辑“自动更正”词条，来添加自己的“自动更正”词条或删除不需要的词条，以达到快速输入的目的。例如：我们可以添加“自动更正”词条，将文档中的ms自动更正为Microsoft。其步骤如下：

（1）打开“自动更正”对话框，如图3-21所示。

（2）在“替换”文本框中输入应被更正的词语，如“ms”，在“替换为”文本框中输入更正后的词语，如“Microsoft”。

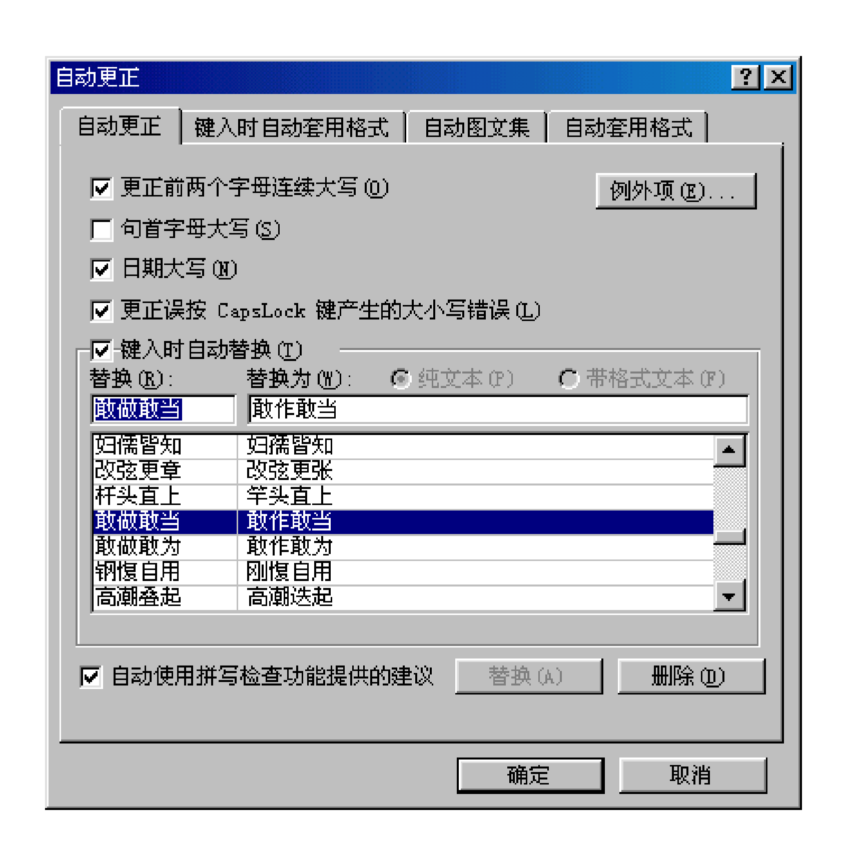


图 3-21　“自动更正”对话框

（3）单击“添加”按钮。

（4）单击“确定”按钮，完成操作。

2. 拼写检查更正

“自动更正”可使用由拼写检查工具主词典生成的更正内容，作为内置的拼写更正列表的补充，从而增强了拼写更正功能。

要使用“拼写检查功能”，只需选中对话框中的“自动选用拼写检查功能提供的建议”选项框即可。

3. 防止“自动更正”做出特殊修改

对于“自动更正”的大写和拼写检查选项，还可创建“例外项列表”，

用于指定不需要“自动更正”进行的更正。例如，可以防止“自动更正”将缩写“eg.”之后键入的单词改为大写。其步骤如下：

（1）打开“自动更正“对话框

（2）选择“例外项”按钮，打开“自动更正例外项“对话框，如图 3-22 所示：

（3）选择“首页面大写”页面标签。

（4）在“其后不大写”文本框内输入“eg.”，单击“添加”按钮。

（5）单击“确定”按钮，完成操作。

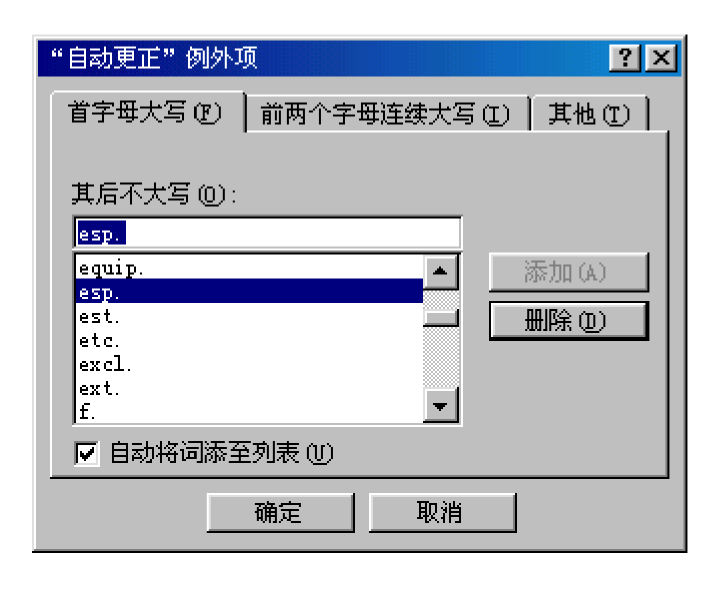


图 3-22　“自动更正例外项“对话框

3.4.2　自动套用格式

Word2000 的自动套用格式功能可以大大减轻常用文档中格式化的工作，并加快了通用文档的格式化工作。在 Word 中“自动套用格式”的用途有：

（1）对文字快速应用标题、项目符号和编号、边框、数字、符号以及分数等格式。

（2）自动将 Internet、网络和电子邮件地址设置为超级链接。

（3）将括在星号 (*) 或下划线 (_) 中的文字更改为粗体或斜体。

（4）将两个连字符 (--) 替换为长划线 (—)。

Word 提供了以下两种自动设置文档格式的方法：

- 在键入的同时自动设置文字格式

如果设置了“自动更正”对话框中“键入时自动套用格式”选项卡上的选项（如图 3-23 所示），那么 Word 2000 会在键入文档的同时，自动设置其格式。例如，如果在数字后键入句号或连字符，再键入空格或制表位，然后键入文字，Word 会将文字设置为编号列表。可以选择在键入时由 Word 自动进行哪些格式更改。

✍ 清除所有选项，可彻底关闭自动设置格式功能。如果不希望完全关闭某选项，而只是不希望在某处使用该功能，那么可以在 Microsoft Word 自动设置格式后，立即单击“撤消”按钮。

- 编写完文档后自动设置文档格式

可以在编辑完文档之后一次性完成所有文本的格式设置，方法是使用“格式”菜单中的“自动套用格式”命令。您可以逐项审阅 Word

进行的所有修改，并决定是接受还是拒绝。通过一次单独操作自动设置文档的格式。

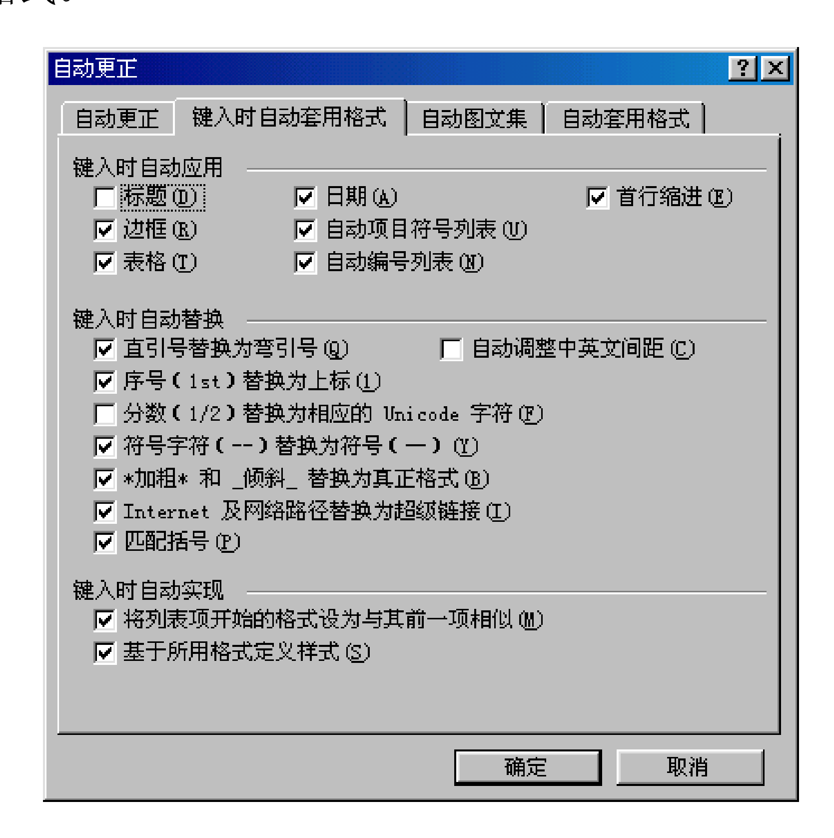


图 3-23　“键入时自动套用格式”选项卡

3.4.3　自动图文集

在编辑文档时，可能要反复地用到一些词语和图片，可以将它们加入到 Word2000 的“自动图文集”，这样，以后再要输入这些词语就非常简单了。

在文档中插入“自动图文集”中的词条的方法如下：

（1）单击“工具”菜单中的“自动更正”命令，打开“自动更正”对话框。

（2）选择如图 3-24 所示的“自动图文集”选项卡。

（3）在列表框中选中要插入文档中的词条，单击“插入”按钮。即可将词条插入到文档中。

要从自动图文集中删除某不常用的词条，只需选定该词条，再单击“删除”按钮即可。

✍ 在“自动图文集”选项界面中，如果选中了“显示有关自动图文集和日期的记忆式输入提示”的话，那么如果输入了自动图文集中某词条的前半部分，Word2000 会自动提示词条的后半部分。例如 Microsoft 是自动图文集中的一个词条，只要在文档中输入 Micro，在光标处就会出现“Microsoft”的提示框，此时按下“F3”或“Enter”键即可输入整个词条。

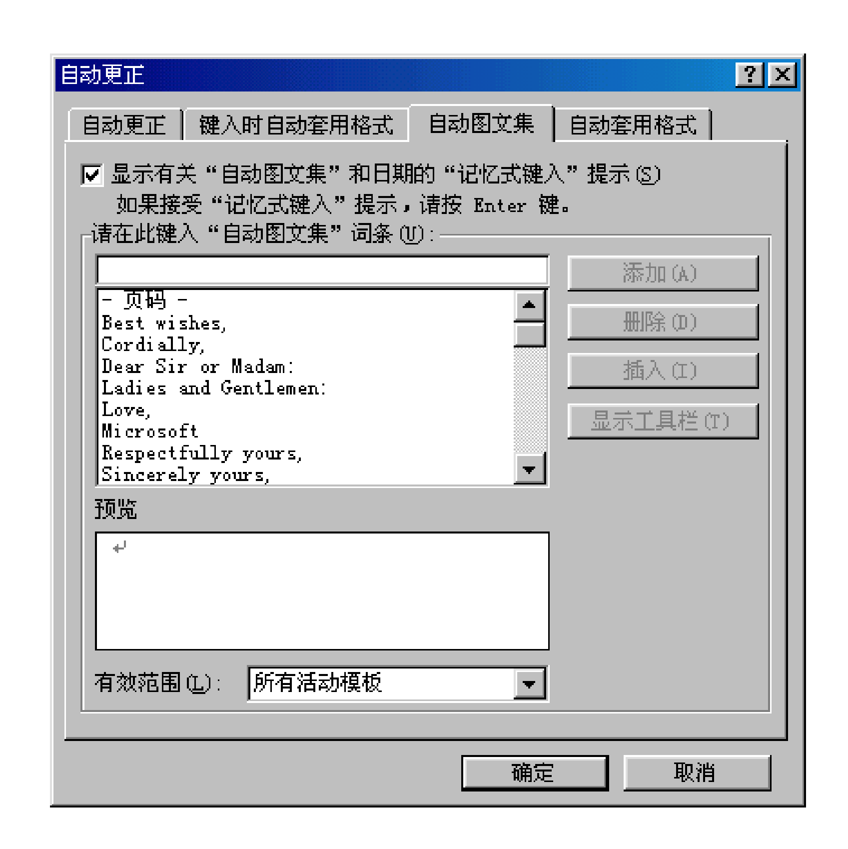


图 3-24　“自动图文集”选项卡

当某些词条或图形需要被经常使用时，可以将其创建为“自动图文集”词条，其步骤如下：

（1）选定要保存为“自动图文集”词条的文本或图形（要在词条中保存段落格式，必须将段落标记也包含在选定内容中）。

（2）单击“插入”菜单“自动图文集”子菜单中的“新建”命令。

（3）在“词条名称”文本框中接受该 Word 给出的缺省名称或键入新的名称。如果要使用“记忆式键入”功能插入词条，应确保名称至少包含四个字符，因为只有在键入四个字符后才能插入词条。

✍ 默认情况下，Word 将“自动图文集”词条保存在 Normal

模板中，以使其可用于所有文档。要只需在特定文档中使用“自动图文集”词条，可指定一个模板用于保存该词条。方法是：选定要保存为“自动图文集”词条的文本或图形，指向“插入”菜单中的“自动图文集”子菜单，单击“自动图文集”命令，再单击“有效范围”框中的模板名称。在“请在此键入‘自动图文集’词条”框中键入该“自动图文集”词条的名称。然后单击“添加”按钮。

3.4.4 自动编写摘要

如果文档的结构很复杂的话，我们可以使用 Word 的“自动编写摘要”功能来自动为文档编写摘要，以方便文档的阅读。

为命令编写摘要的步骤如下：

（1）单击“工具”菜单中的“自动编写摘要”命令。

（2）打开的“自动编写摘要”如图 3-25 所示，指定摘要的类型和摘要相当于原长的百分比。

（3）单击确定按钮。

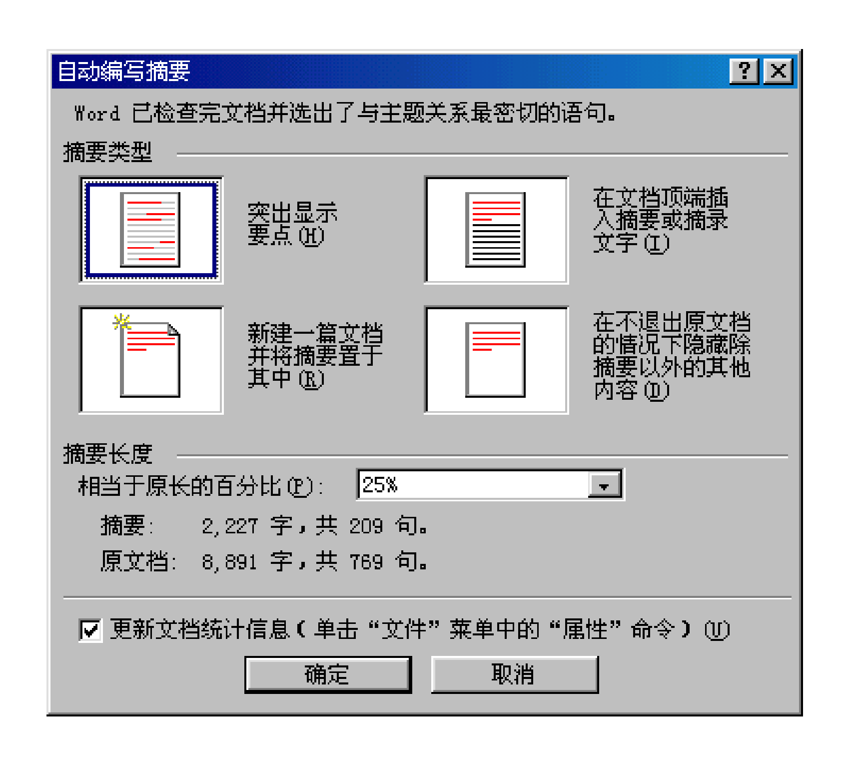


图 3-25 “自动编写摘要”对话框

由上图可见，摘要的类型有如下四种：

- 在文档中突出显示要点。
- 在文档顶端插入摘要或摘录文字。
- 新建一片文档并将摘要置于其中。
- 在不退出原文档的情况下隐藏除摘要外的其他内容。

3.5 保护文档

在第一部分的最后，将对如何保护文档进行简要的介绍。

学会保护自己的文档是很重要的。一方面，文档经常可能遭到一些意外情况的破坏，为此应当有一些预防措施，在意外情况发生之后

也应当有一些补救措施；另一方面，有时文档中可能有一些内容是用户不希望他人看到的，或者是不希望他人随意改动的。因此对文档应该有相应的保护措施。本节将介绍一些常用的保护文档的方法。

3.5.1　自动保存

常常有这样一些情况，当 Word 2000 非正常关闭或者计算机突然掉电等意外发生时，用户往往并没有保存正在编辑的文档，或者对文档进行很多修改后还没有存盘，这样的话很多工作就白做了。针对这一问题，Word 2000 提供了“自动保存”功能。

打开“文件”菜单并选择其中的“另存为”命令可打开“另存为”对话框，单击对话框右上角的“工具”按钮，并从其下拉菜单中选择“常规选项”命令，此时将出现一个“保存”对话框，如图 3-26 所示。选中“自动保存时间间隔”复选框，并设置好两次自动保存之间的时间间隔后，单击“确定”按钮即可。

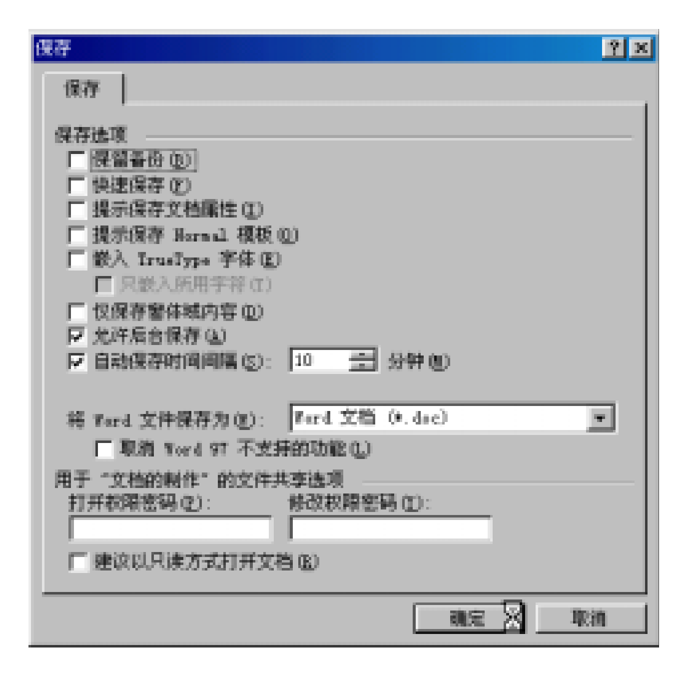


图 3-26　“保存”对话框

启动自动保存功能以后，Word 2000 将周期性地对文档进行保存，并存储在一个独立的临时恢复文件中。此文件扩展名为 Asd，但文件名与文档本来的文件名并不相同。这样即使遇上 Word 非正常关闭或者计算机突然掉电等意外情况，文档也可以还原到上次自动保存时的状态。只要重新启动 Word 2000，所有非正常关闭的文档将会被自动打开，并在标题栏的文档名后显示“(恢复)”字样，实际上它们就是所最后一次自动保存下来的恢复文件。这样就可以把它们重新储存下来。

✍ 应当注意“自动恢复”功能并不等同于正常的文档保存。倘若 Word 打开恢复文件后选择了不进行保存，则该文件将被删除，未保存的修改也将会丢失。而如果选择了

保存恢复文件，它将会替换原文档（除非给它指定了一个新的文件名）。

3.5.2 自动备份

在对文档进行处理时，常常会在不小心删除了文件中某些有用的内容后进行了保存。为了减少这种情况带来的损失，Word 2000 提供了自动备份功能，在每次保存文档时自动保留一个备份。该备份提供了上一次所保存的副本，这样，文档中保存有当前所保存的信息，而副本中保存有上次所保存的信息。

启动自动备份功能的方法是，按照前面的方法打开如图 3-26 所示的“保存”对话框，选定“保留备份”选项后，单击“确定”按钮即可。

自动备份功能启动之后，Word 2000 将在每次保存文档时把文档的前一个版本作为备份文件保存起来。这个备份文件与文档本身的文件名相同，但扩展名为 Bak。

✍ 当 Word 2000 启动自动备份功能之后，在每次保存文档时，新的备份将会取代已有的备份。

3.5.3 口令保护

如前所述，有时候用户不希望他人看到自己的文档，或者不希望他人对自己的文档随意进行改动。为此，Word 2000 提供了口令保护

功能。用户可以对文档设置口令，以保证自己的文档不被他人看到或更改。

对当前文档设置保护口令的方法是，按照前面的方法打开图 3-26 所示的“保存”对话框，在“打开权限密码”或者“修改权限密码”框中键入用户设置的口令后，单击“确定”按钮即可。口令可以由 15 个字符以下的字符串组成，其中可以包括字母（区分大小写）、数字、符号和空格。

“打开权限密码”用以防止不知道口令的人看到文档。设置该密码后，在下一次打开文档时将出现类似于图 3-27 所示的对话框，只有键入正确的密码才能打开文档并对其进行修改和保存。



图 3-27　打开权限密码对话框

“修改权限密码”用以防止不知道口令的人对文档进行改动。设置该密码后，在下一次打开文档时将出现类似于图 3-28 所示的对话框，只有键入正确的密码才能对文档进行修改并将修改保存下来，否则的话只能以只读方式打开文档。

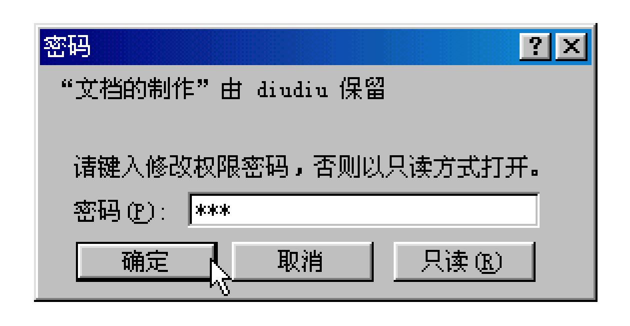


图 3-28　修改权限密码对话框

在图 3-26 所示的“保存”对话框中还有一个“建议以只读方式打开文档”复选框，如果选中这一项，则下一次打开文档时将出现类似于图 3-29 所示的对话框。若希望以只读方式打开文档，单击“是”按钮；若希望打开文档并进行修改和保存，则单击“否”按钮；“取消”按钮表示不打开文档。用户可以利用这项功能提醒自己或他人不要修改某个文档。



图 3-29　建议以只读方式打开文档对话框

3.5.4　保存多个版本

在有时候，用户需要多次对文档作适当的修改，并先后形成不同的版本，而这些版本对用户来讲可能都是有用的。Word 2000 提供了一种多版本保存的功能，当用户想纪录文档的改变时，可以在同一文档中保存文档的多个版本。在这种功能下，Word 2000 将只保存文档

的一个保存的版本与其下一个版本之间的差异。保存了文档的几个版本之后，可以回过头来审阅、打开、打印或者删除以前的版本。如果某各版本遭到了破坏，还可以通过修改相似的版本来补救。

打开菜单栏中的“文件”菜单，并选择其中的“版本”命令，将出现一个“版本”对话框，如图 3-30 所示。在这里可以打开或删除某个版本，也可以查看各个版本的备注。把当前文档保存为一个版本的方法是，单击“现在保存”按钮，此时将出现一个“保存版本”对话框，如图 3-31 所示。在“版本备注”一栏中键入关于正在保存的版本的说明后，单击“确定”按钮即可。



图 3-30　“版本”对话框

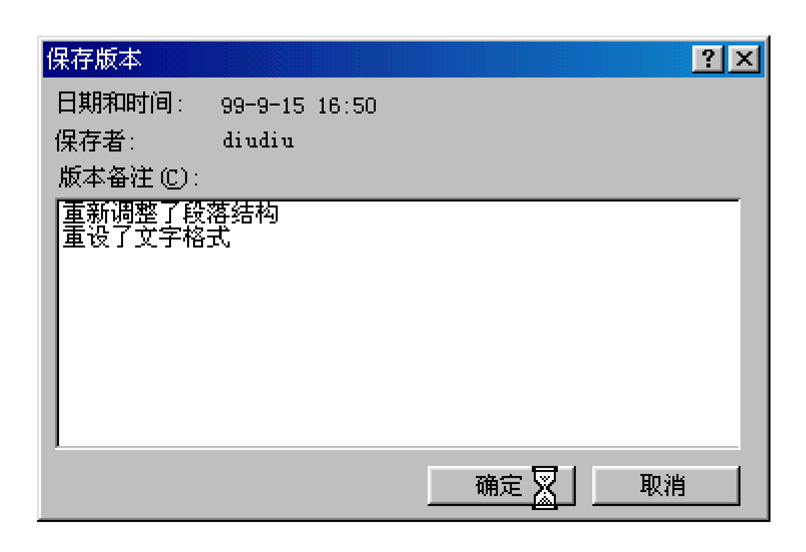


图 3-31 “保存版本”对话框

有时用户需要记录文档修订的作者和时间（例如对于法律文档），这时可以利用关闭文档时自动保存版本功能。方法是打开图 3-30 所示的“版本”对话框后，选中“关闭时自动保存版本”复选框，再单击“关闭”按钮即可。

第四章　VBA 的词法和语法

VBA 语言是建立在 Visual Basic 语言基础之上的，它的词法和语法跟 VISUAL BASIC 也是基本相同的。本章将主要介绍 VBA 语言的数据类型和控制结构。

4.1　数据类型

VBA 的数据类型包括布尔型（Boolean）、日期型（Date）、字符串（String）、货币型（Currency）、小数型（Decimal）、字节型（Byte）、整数型（Integer）、长整数型（Long）、单精度浮点型（Single）、双精度浮点型（Double）、对象（Object）、用户自定义型及变体（Variant）。变量是存储数据的所在处，每个变量都有名字和数据类型。变量的数据类型决定了如何将代表这些值的位存储到计算机的内存中去。根据缺省规定，如果在声明中没有说明变量的数据类型，则变量的数据类型为 Variant。Variant 类型可在不同场合代表不同数据类型，而且，当指定变量为 Variant 时，不必进行数据类型的转换，Visual Basic 会自动转换。下面将分别详述以上各种数据类型的存储长度、表示范围、类型声明字符以及相互间的转换，并给出某些简单示例。

4.1.1　布尔型

布尔型（Boolean）变量占用 2 个字节（16 位），取值只能是 True 或 False。缺省值是 False。如果变量的值只是“True/False”、“yes/no”、“on/off”等信息，则可将其声明为 Boolean 型。

下面的例子中，变量 diskrun 是 Boolean 型的，用来存储简单的 yes/no 设置。

```
Dim　diskrun　As Boolean
'查看磁盘是否在转，转为 1，不转为 0。
    If　Recorder.Direcrion=1　Then
    Diskrun=True
End　if
```

当转换其它的数据类型为 Boolean 值时，0 会转成 False，而其它的值则变成 True。当转换 Boolean 值为其它的数据类型时，False 成为 0，而 True 成为-1。

4.1.2　日期型

VBA 提供了一个存储日期和时间值的数据类型—Date 型。它简洁方便，容易掌握，但要熟练使用日期值，仍然需要一些技巧，首先需要了解 VBA 内部是如何存储日期值的。

日期型（Date 型）变量占用 8 个字节，浮点数值形式，所以在系

统内部与双精度浮点型（Double 型）相同。它可以表示的日期范围从 100 年 1 月 1 日到 9999 年 12 月 31 日，而时间可以从 0:00:00 到 23:59:59。

VBA 在 Date 变量的整数部分存储日期值，在小数部分存储时间值。日期的小数部分表示从子夜到现在已经度过的时间。如果小数部分是.50，它表示一天已过了二分之一，现在的时间是正午 12 点。1899 年 12 月 31 日之前的日期以负整数表示，该日期之后的日期才是正整数，如果将 1899 年 12 月 31 日的日期转换成双精度浮点型(Double)的数，会看到该日期是零值。当其它的数据类型要转换为 Date 型时，小数点左边的值表示日期信息，而小数点右边的值则表示时间，午夜为 0 而中午为 0.5。

日期文字应该用数字符号#扩起来，例如下面定义日期型的变量并赋值：

```
Dim dbToday As Date
'将 dbToday 这个变量定义为 Date 型变量。
dbToday=#September 12, 1999#
'或写成 #12 Sep 99#的形式，作用相同。
```

Date 变量会根据计算机中的短日期格式来显示。时间则根据计算机的时间格式（12 或 24 小时制）来显示。任何可辨认的文本日期都可以赋值给 Date 变量。

此外，VBA 中用于获取当前日期和时间的函数有 Now，Date 和

Time。

调用 Now 函数可以同时返回 Date 型变量的日期和时间。

举例来说：

在立即窗口（Immediate Window）调用 Now 函数，将返回类似于下面这种形式的值：

9/14/1999 6:21:05PM

调用 Date 函数只返回当前日期，不显示当前时间。

调用 Time 函数可以返回当前时间，但不包括当前的日期。

✍ 注意：当在代码中使用文本日期时，必须告知 VBA 该值是日期值；否则，很可能认为在做浮点除法。例如，在下面的代码中，VBA 赋给 Date 变量的值不是 1999 年 9 月 14 日，而是 9 除以 14 除以 1999。

```
Dim dbDate As Date
dbDate=9/14/1999
Debug.Print dbDate
```

所以，VBA 的日期分割符是 #，如果使用日期分割符重写上述例子，就不会误认了。

```
Dim dbDate As Date
DbDate=#9/14/1999#
Debug.Print dbDate
```

- Microsoft Office 2000 支持的日期都跨越了 2000 年，但是，在正确的设计和处理二十世纪和二十一世纪的日期时，还应注意一些问题。在 Microsoft Year 2000 Resource Center Web 站点中有详尽介绍，地址是：http://www.microsoft.com/technet/topics/year2k/default.htm

4.1.3　字符串型

字符串型（String）变量分为固定长和可变长两种。

变长字符串最多可包含大约两兆个字符。定长字符串最多可包含大约 64K 个字符。

- 注意 Public 定长字符串不能在类模块中使用。所谓类模块，即含有类定义的模块，包括其属性和方法的定义。除非受 Option Private Module 的影响，否则，使用 Public 语句声明的变量对所有应用程序的所有模块中的所有过程都是可见的。在 Option Private Module 的影响下，变量仅在包含它的工程中为公共的。

String 的字符码的范围是 0 到 255。字符集的前 128 个字符（0 到 127）对应于标准的 U.S.键盘上的字符与符号。这前 128 个字符与 ASCII 码字符集中所定义的相同。后 128 个字符（128 到 255）则代表特殊字符，例如国际字符，重音符号，货币符号及分数。String 的类型声明

字符为美元号($)。

将包含字符串而从不包含数值的变量声明为字符串型的例子：

```
Private  S  As  String
S="Database"
S=Left(S，4)
```

这是先将字符串“Database”赋予变量 S，再用字符串函数对它进行操作。按照缺省规定，String 变量或参数是一个可变长度的字符串，随着对字符串赋新值，变量的长度就可增可减。

也可以声明字符串具有固定长度。可用下面的语法声明一个定长字符串：

```
String  *size
```

例如：

```
Dim  AuthorName  As  string*20
```

是声明一个长为 20 字符的字符串。如果长度小于 20 个，则用空格将 AuthorNamed 的剩余部分添满，如果赋予字符串长度太长，超过 20 个，已不能成为定长字符串，那么 Visual Basic 会直接截去多余部分的字符。

另外，因为定长字符串用空格填充尾部的多余空间，所以，在处理定长字符串时，删除空格的 Trim 和 Rtrim 函数特别有用。在标准模块中，定长字符串可以声明为 Public 或 Private 型的，但是，在窗体和

类模块中，必须将定长字符串定义为 Private。

如果字符串表示数值，可以将字符串赋予数值变量，也可将数值赋予字符串变量。例如，将命令按钮、文本框、列表框放置于窗体中，然后在命令按钮 Click 事件中输入代码如下：

```
Private  Sub  Command 1_Click  ()
    Dim  intX  As  Integer
    Dim  strY  As  String
    strY=“11.77”
    intX=strY
    '将字符串传递给数值变量。
    List1.AddItem Cos(strY)
    '将字符串中的数值的余弦值添加到列表框中。
    StrY=Cos(strY)
    '又将余弦值传递给字符串变量。
    Text1.Text=strY
    '在文本框中显示字符串。
End  Sub
```

一般来说，Visual Basic 会自动强制变量为适当的数据类型。在转换字符串和数值时要谨慎，因为如果传递的字符串的值不是数值，在运行时就会出错。

4.1.4 货币型

货币型变量（Currency）占用 8 个字节，整型的数值形式。它与下面介绍的小数型（Decimal）属于 VBA 中的两种比例数据模型，它们提供了高标准的精确度，这些类型也被称为定点数据类型。但是，它们没有浮点型精确，也就是说，它们无法像浮点类型那样表示过大或过小的值。如果不能容忍舍入误差，且不需像浮点类型那样提供大范围和小数点，那么，可以使用定点整型的数据类型。

比例整型的数据类型采用一个整数乘以 10 的因子来表示。可以表示小数点左边的 15 位数字，右边的 4 位数字。这种表示法的范围可以从 -922,337,203,685,477.5808 到 922,337,203,685,477.5807。

Currency 数据类型在货币计算与定点计算中很有用，因为在这种场合精度特别重要。Currency 的类型声明字符为 at 号 **@**。

4.1.5 小数型

Decimal 型变量占用 12 个字节（96 位），带符号的整型形式，是将给定数值除以一个 10 的某一个幂数所得到的。这个幂数即为变比因子，它决定了小数点右面的数字位数，其范围是从 0 到 28。当变比因子为 0，即给定数值除以 10 的零次幂，最大的可能值为 +/-79,228,162,514,264,337,593,543,950,335。而在有 28 个小数位的情况下，最大值为 +/-7.9228162514264337593543950335，小数型

（ Decimal ） 的 数 据 类 型 最 小 的 非 零 值 为 +/-0.0000000000000000000000000001。

✍ 注意：此时 Decimal 数据类型只能在 Variant 中使用，也就是说，不能声明一变量为 Decimal 的类型。事实上，Decimal 数据类型是 Variant 的子类型。因此，如果要使用 Decimal 类型，必须先声明一个 Variant 变量，然后用 Cdec 函数进行转换。

下面的例子不仅给出怎样将 Variant 变量转化为 Decimal，而且告诉我们使用 Decimal 类型的变量可以减小浮点数据的固有舍入误差。

```
Sub DoubleVsDecimal()
    Dim dbX As Double
    Dim dbY As Variant
    Dim Count As Long
'以上是声明 dbX 是双精度浮点型变量，dbY 是 Variant 类型变量，而 Count 是长整型变量。
   For Count=1 To 100000
    dbX=dbX+1
    dbY=dbY+Cdec(0.00001)
    '这一步是将 Double 型变量通过函数 Cdec 转化成 Decimal 型，并进行运算。
```

```
    …
    Next
    …
    Debug.Print"Result in Double："&dbX
    Debug.Print"Result in Decimal："&dbY
End Sub
```

上述过程在立即窗口（Immediate Window）中输出的结果是：

```
Result in Double：0.999999999998084
Result in Decimal：1
```

可见，使用 Decimal 类型的变量的确可以降低浮点数据类型固有的舍入误差。

4.1.6 字节型

Byte 类型的变量存储的是单精度型、无符号整型、8 位（1 个字节）的数值，范围在 0 至 255 之间。

Byte 这种数据类型在存储二进制数据时很有用。因为在转换格式期间，用 Byte 型的变量存储二进制数据就可以保留数据。如果用 String 变量在 ANSI 和 Unicode 格式间进行转换，那么变量中的任何二进制数据都会遭到破坏。所以，如果变量包含二进制，则应把它声明为 Byte 数据类型的数组。关于数组的详细内容将在本书后面介绍。Byte 类型

的变量的除了保存数据，主要作用是进行二进制数据的保存，以便与其它 DLL 或 OLE Automation 对象联系。

因为 Byte 类型是 0~255 范围的无符号类型，所以不能表示负数。因此，除了一元减法外，所有可对整数进行操作的运算符均可操作 Byte 数据类型。而在进行一元减法运算时，Visual Basic 先将 Byte 类型转换成符号整数。

另外，所有数值变量都可以相互赋值，也可以对 Variant 变量赋值。在将浮点数赋予整数之前，Visual Basic 将浮点型的小数部分四舍五入，而不是简单的省掉小数部分。

声明 Byte 类型的变量的方式如下：

Dim ByteA As Byte

ByteA 就是一个字节型变量。

事实上，Byte 数据类型经常用于操作字符串，对某些字符串操作来说，将字符串转换成字节数组可以大大提高运算性能。

4.1.7 整数型

Integer 类型的的变量存储 16 位（2 个字节）的数值，其范围为 -32,768 到 32,767 之间。声明整数类型变量时，可以用 Integer 关键字，也可以使用整数类型声明字符—百分比符号 (%)。例如：

Dim IntA As Integer

Dim　IntB%

其中，IntA 及 IntB 都是 Integer 类型的变量。

传统上，VBA 程序员使用 Integer 保存较小数值的整数，因为它们占用内存少。但是，现在的 VBA 版本将所有 Integer 值转换成 Long 类型，即使变量已被声明成 Integer 型。使用 Integer 型变量在速度上不再有什么优越性，因为 Long 类型甚至可能比 Integr 类型稍微快一些，原因是 VBA 不必再做类型之间的转换。

如果知道变量总是存放整数，不带有小数点，就应该将它声明为 Integer 或 Long 类型。整数的运算速度较快，占用的内存空间通常较小。尤其在 For…Next 循环中作为计数器变量使用时，Integer 或 Long 类型就很有用。

也可以用 Integer 变量来表示枚举值。枚举值可包含一个有限集合，该集合包含的元素都是唯一的整数，每一个整数都在它使用时的上下文当中有其特殊意义。枚举值为在已知数量的选项中做出选择提供了一种方便的方法，例如，black = 0，white = 1 等等。较好的编程作法是使用 Const 语句将每个枚举值定义成常数（即执行程序时保持常数值的命名项目，可以是字符串、数值、另一常数、大部分算术运算符或逻辑运算符的组合）。

4.1.8　长整数型

Long（长整型）类型变量存储 32 位（4 个字节）有符号的数值，其范围从 -2,147,483,648 到 2,147,483,647。Long 的类型声明字符为和号(&)。与 Integer 类型一样，Long 类型变量可以用 Long 关键字来声明，也可以用“&”说明。

例如：

```
Dim　LongIntC　As　Long
```

4.1.9　单精度浮点型

Single（单精度浮点型 Single-precision floating-point）类型变量存储 32 位（4 个字节）的浮点数值。它的范围：

负数：从 -3.402823E38 到 -1.401298E-45

正数：从 1.401298E-45 到 3.402823E38

零　：0。

Single 的类型声明字符为感叹号“！”。

✍ 注意：Single 类型表示的数据类型并不是精确的，所以，如果程序中所用数值不是很大，应该避免使用浮点类型的变量。

下面举例声明单精度浮点变量：

```
Dim　S!，S01　As　Single
```

S 与 S01 均为单精度类型变量。

4.1.10　双精度浮点型

Double（双精度浮点型 Double-precision floating-point）类型变量存储 64 位（8 个字节）浮点数值。它的范围是：

负数：从 -1.79769313486231E308 到 -4.94065645841247E-324

正数：从 4.94065645841247E-324 到 1.79769313486232E308（包括零）。

Double 的类型声明字符是符号#。它在保存数值时的有效位数比 Single 大得多，而且可以表示较大的数值。

但是，在 Single 或 Double 类型的有效范围内，不是所有的数值都可以用二进制形式表示出来，所以产生了截断。浮点计算存在固有局限——用二进制数值表示十进制数值。此外，某些数值不能用有限位小数表示，如圆周率 pi，1/6 的小数形式等等。由于这些限制，在执行浮点运算时，舍入误差不可避免。当精确度要求不高，用浮点类型表示特别大或特别小的数值是非常合适的。但是，如果所用数值必须精确，就要考虑几种定点整型数据类型之一，比如，处理货币值时。

从 1.1.6 到 1.1.10 介绍了 Integer 型、Long 型、单精度浮点型、双精度浮点型四种数据类型，它们是数值型（Numeric）的四种子类型。

4.1.11　对象

对象（Object）型变量用 32 位（4 个字节）的地址来存储。该地址可以引用应用程序中或其它某些程序中的对象。利用 Set 语句，声明一个 Object 的变量，可以引用应用程序中的任何实际对象。例如：

```
Dim　objDb　As　Object
Set　 objDb=OpenDatabase("c:\Visual Basic\Bible.mdb")
'这是先声明 objDb 为对象型变量,再用 Set 语句引用路径为 c 盘下 Visual Basic 目录下的'Biblio.mdb 中的 OpenDatabase 对象。
```

在声明对象变量时，应该使用特定的类，而不用一般的 Object。比如上面的例子中，用 Database 取代 Object。运行应用程序之前，Visual Basic 可以决定引用特定类型的对象的属性和方法。这样，程序运行速度会更快。关于特定的类，在“对象浏览器”中列出。

> 注意：虽然以 Object 类型声明的变量可以用于引用各种对象，但是绑定到变量引用的对象总是在晚期(即运行时)绑定。若要强迫在早期（即编译时）绑定的话，须将引用的对象赋值给以特定类名称声明的变量。

4.1.12　用户自定义型

用户自定义类型是用 Type 语句定义的数据类型,可以包含一个或多个某种数据类型的数据元素、数组或一个先前定义的用户自定义类

型。例如：

```
Type Myrecord
    MyName As String
    '定义字符串变量存储名字。
    Mymerrage As Boolean
    '定义布尔变量存储婚姻状况（0为未婚，1为已婚）。
    Mybirth As Date
    '定义日期变量存储出生日期。
End  Type
```

4.1.13　变体

Variant 数据类型是所有没被显式声明（用如 Dim、Private、Public 或 Static 等语句）为其它类型变量的数据类型。Variant 数据类型并没有类型声明字符。如果定义变量时缺省“As 类型”部分，则为变体型变量。

Variant 是一种特殊的数据类型，除了定长字符串数据(1.1.3)及用户定义类型(1.1.13)外，可以包含任何种类的数据。因此变体类型的变量可以说是 VISUAL BASIC 中应用最广泛、最灵活的一种数据类型。变体型变量不仅可以存储所有类型的数据，而且当赋予不同类型值时，可以自动进行类型转换。例如：

Dim Thing

'默认为变体型。

Thing="20"

'赋值为字符串“20”（双字符的串）。

Thing=Thing-14

'现在包含为数值 6。

Thing="B"&Thing

'现在为双字符串“B6”。

Variant 还包含 Empty、Error、及 Null 三种特殊值。下面介绍常用的 Empty、Null、Error 这三个特别的值。

Empty：在给变体型变量赋值前具有的值。用数值表示时，以 0 表示 Empty 变量；在字符串表示时，则以零长度字符串("")表示它。但它是异于 0，零长度字符串("")、Null 值的特定值。通常用 IsEmpty 函数来测试 Empty 的值：

If　IsEmpty(Z)

Then　Z=0

当 Variant 变量包含 Empty 值时，就可以在表达式中使用它。只要将任何值（包括 0，零长度字符串和 Null）赋给 Variant，Empty 值就会消失。而将关键字 Empty 赋给 Variant 变量，就可将 Variant 变量恢复成 Empty。

Null：经常用于数据库应用程序中，表示未知或丢失的数据。不应将 Null 与 Empty 弄混。Empty 值用来标记尚未初始化（即给定初始值）的 Variant 变量；而 Null 是表示 Variant 变量确实含有一个无效数据。

由于在数据库中使用 Null 值，所以 Null 具有某些唯一的特性：

（1）对包含 Null 的表达式，计算结果总是 Null。

（2）将 Null 值、含 Null 的 Variant 变量、计算结果为 Null 的表达式作为参数传递给大多数函数，函数返回值为 Null。

（3）Null 值由返回 Variant 数据类型的内在函数传播。

也可以用 Null 关键字定义 Null：

```
Z=Null
```

也可以用 IsNull 函数测试 Variant 变量是否包含 Null 值：

```
If   IsNullL(X) And IsNull(Y) Then
    Z=Null
Else
    Z=0
End   If
```

另外，如果不在应用程序中使用 Null 值，就不必书写测试 Null 和处理 Null 的程序。这是因为，除非明确规定将 Null

值赋予变量，否则变量不会设置成 Null 值。

Error：用于指出已发生过程中的错误状态。一般种类的错误，一旦出错，程序产生普通的应用程序级的错误处理。但是对 Error 这个特殊值，程序并不产生正常的应用程序级的错误处理。这可以让程序员，或应用程序本身，根据此错误值采取另外的行动。可以用 CVErr 函数将实数转换为错误值来产生 Error 值。

以上介绍了三种 Variant 的特殊值，详细资料请参阅『联机手册』的“语言参考”部分中“Null”“CVErr”和“Empty”等。

当 Variant 表示数值数据时，可以是任何整型或实型数，负数时范围从-1.797693134862315E308 到-4.94066E-324，正数时范围则从 4.94066E-324 到 1.797693134862315E308。

通常，数值 Variant 数据保持为其 Variant 中原来的数据类型。例如，如果把一个 Integer 赋值给 Variant，则接下来的运算会把此 Variant 当成 Integer 来处理。然而，如果算术运数针对含 Byte、Integer、Long 或 String 之一的 Variant 执行，并当结果超过原来数据类型的正常范围时，则在 Variant 中的结果会提升到较大的数据类型。如 Byte 则提升到 Integer，Integer 则提升到 Long，而 Long 和 Single 则提升为 Double。当 Variant 变量中的 Currency、Decimal 及 Double 值超过它们各自的范围时，会发生错误。

可以用 VarType 函数或 TypeName 函数来决定如何处理 Variant 中的数据。

VarType()函数返回的值及意义如表 4-1 所示。

表 4-1　VarType 函数的返回值及意义

返回值	意义
0	这是一个 Variant 类型变量还不曾被存入任何值(Empty）
1	Variant 类型变量中所存放的数据不合法（Null）
2	Variant 类型变量中所存放的是整数类型（Integer）
3	Variant 类型变量中所存放的是长整数类型（Long）
4	Variant 类型变量中所存放的是单精度浮点类型（Single）
5	Variant 类型变量中所存放的是双精度浮点类型（Double）
6	Variant 类型变量中所存放的是货币类型（Currency）
7	Variant 类型变量中所存放的是日期类型（Date）
8	Variant 类型变量中所存放的字符串类型（String）
9	Variant 类型变量中所存放的是对象（Object）
10	Variant 类型变量中所存放的是错误代码（Error Value）
11	Variant 类型变量中所存放的是布尔类型（Boolean）
12	Variant 类型变量中所存放的是 Variant 数组（Variant Array）
13	Variant 类型变量中所存放的是数据处理对象（DAO）
14	Variant 类型变量中所存放的是 Decimal 类型(Decimal）
17	Variant 类型变量中所存放的是字节类型（Byte）
8192	Variant 类型变量中所存放的是一个数组（Array）

以上的十种基本类型，再加上用户自定义型(User-defined Type)、对象类型(Object Type)和附属于 Variant 类型的 Decimal 类型，总共是十三种类型的 Visual Basic 数据。

4.2　常量

4.2.1 常量的作用

经常会发现程序代码包含一些固定的数值或字符串，且这些数值或字符串在程序中反复出现，为这些数值或者字符串起一个有意义的名称，即定义为常量，则可以大大改进代码的可读性和可维护性。

将固定数值或字符串定义为常量的优点有：

（1）提高程序可读性。

举例来说，在程序中有一个数字未命名，那么可能必须花费一段时间猜想它到底是干什么用的。如果以一个常量命名它，比如 Bank_Interest，那么就不必知道 Bank_Interest 是何值，只要知道它代表银行利息就足够了。可见，定义了常量可以提高程序的可读性。

（2）使程序易于修改。

再以前面的例子为基础。如果要写一个银行计算利息的程序，必然多次用到利率，昨天的利率与今天不同，是否要一行一行的修改昨天的程序呢？如果定义利率常量，只要在程序开头修改这个常量的定义，就可以实现修改。

（3）减少出错机率。

如果利率是 0.00456789,在反复书写中极易出错，如定义为常量 Interest_Rate，避免输入错误。

4.2.2 常量的来源

常量一般有两种来源：

（1）程序和控件提供内部的或系统定义的常量。VBA 中有内部定义的常量，如 Visual BasicOk，Visual BasicYes，Visual BasicNo。

（2）用 Const 语句来声明自己的常量。

在 Visual Basic 中，常量名可以采用大小写混合的形式。设计名字时应该尽力防止发生意外冲突，不能出现名字相同但表示不同数值的情况。

4.2.3 定义常量

它实际上是常量的第二种来源，即用 Const 语句声明常量。语法格式为：

[Public|Private]Const 常量名称 [As 数据类型]=表达式

定义常量时，可以替常量加入数据类型说明，如 As String，As Double 等，但通常很少这麽用，因为意义不太大。

Const 语句可以表示数量或 Date/Time 量。

例如：

```
Const Pi As Single=3.1415926
'声明 Pi 为单精度型的常量 3.1415926。
Public Const Ok As Double=6.08E-3
```

'声明 Ok 为双精度型的常量 6.08×10^{-3}

Const SomeVal As Integer=19

'将整型常量 SomeVal 声明为 19。

Private Const SomeDate=#9/13/99#

'将日期型常量 SomeDate 声明为 99 年 9 月 13 日。

Const 语句也可以定义字符串常数，例如：

SomeName As String=“David”

'将字符型常量 SomeName 声明为 David。

如果在一行中声明几个常量，要用“,”分开；如果分行，要使用续行符“_”。例如：

Public Const Date=#1/1/2000#，Author=“Nancy”，_

Tel=“62774059”

可见等号（＝）右边可以是结果为数字或字符串的表达式（不包括函数调用）。甚至还可以是用先前定义过的常数再定义新常数。例如：

Const Pi2=Pi*10

一旦定义了常数，就可将其放置于代码中，使代码更可读。例如：

Static SolarSystem(1 To SomeVal)

If Pi2>OtherVal Then Exit Sub

至于常量的第一种来源，即应用程序和控件提供内部的或系统定义的常量，通常在联机帮助中有详细介绍。

4.2.4 常量定义范围的规则

常量有它们自己的作用域，与变量声明一样，Const 语句也有范围，也使用一定规则。

（1）仅在程序内部过程中创建的常量，只能在程序中使用。

（2）为创建在整个范围中有效的常数，请在标准模块（.BAS 模块）的声明段中进行声明，在 Const 前面放置关键字 Public。注意在窗体模块或类模块中不能声明 Public 常数。

（3）在 Form 的说明区中定义的常量，只能用 Const 或 PrivateConst 定义，而且只能在 Form 中的各个程序中应用，对模块以外的任何代码都无用。

4.2.5 避免循环引用

由于常数可以用其它常数定义，因此在两个常数之间，注意不要出现循环引用。如果程序中有两个以上的公用常量，而且每个公用常量都用另一个去定义，就出现了循环。举例说明：

'在模块 1 中：

Public Const TotalA=TotalB+1'在整个应用程序中有用。

'在模块 2 中：

Public Const TotalB=TotalA-1'在整个应用程序中有用。

如果出现类似上面这种循环情形，试图运行程序时，Visual Basic

就会告知程序出错，不解决循环问题就不能运行程序。为避免出现循环，建议将公共常数限制在单一模块内，或最多只存在少数几个模块内，以方便管理，减少错误。

4.3　变量

本节将介绍在 Visual Basic 中用来临时存储数值的变量，详细叙述在声明变量、存储和检索变量、变量的有效作用范围、静态变量、数组变量等方面的内容。

4.3.1 声明变量

声明变量就是事先将变量通知给程序。一般来说，用关键字 Dim 语句来声明：

Dim　变量名称　[As　数据类型或者对象类型]

Dim 语句可以被定义为数据类型，如 String、Currency、Integer 类型；也可以是包含来自 Visual Basic 或者其它应用程序的对象，如 Object1，Form1，TextBox 等等。

变量名称应该满足：

（1）以字母开头。

（2）不能包含嵌入的句号或者嵌入的类型声明字符。

（3）在同一范围内必须是唯一的，所谓范围，就是象一个窗体、

一个过程那样的可以引用变量的变化域。

声明变量还有其它形式：

（1）不在过程内部，而在窗体、标准、类模块的声明段中声明变量，这使变量对模块中所用过程都有效。

（2）用关键字 Public 声明变量，这使变量在整个程序中有效。

（3）用关键字 Static 声明一个局部变量，即使过程结束，变量的值也仍然保留。

下面介绍隐式声明和显式声明。

在使用一个变量之前，可以先不声明这个变量，这叫隐式声明。例如书写这样一个函数，就不必在使用变量 empVal 之前声明它：

```
Function  SafeSqr(num)
    TempVal=Abs(num)
    SafeSqr=Sqr(TempVal)
End  Function
```

Visual Basic 用这个名字自动创建一个变量，然后认为它就是显式声明的。虽然这种方法很方便，但是一旦变量名拼写出错，会导致一个很难查找的错误。例如，现在写了这样一个函数：

```
Function  SafeSqr(num)
    TempVal=Abs(num)
    SafeSqr=Sqr(TemVal)
```

```
End Function
```

两段代码似乎一样，但是在第 4 行将 TempVal 拼写成 TemVal，所以函数值总是返回 0。事实上，当 Visual Basic 遇到新名字时，它分辨不出这是意味着声明了一个新变量呢，还是拼写上的错误。于是，只好用这个名字再创建一个新变量。

关于显式声明，就是凡是遇到一个未经明确声明的变量名，Visual Basic 就发出错误警告。

在类模块、窗体模块或标准模块中的声明段加入这样的语句：

```
Option Explicit
```

在“工具”中选“选项”，单击“编辑器”选项卡，再复选“要求变量声明”这一项，这样就会在任何新模块中自动插入 Option Explicit 语句，就可以对此类问题显示错误信息。再举上一段隐式声明的例子，如果拼写发生错误，写成：

```
Function SafeSqr(num)
    TempVal=Abs(num)
    SafeSqr=Sqr(TemVal)
End Function
```

那么，Visual Basic 就会对拼错的变量名 TemVal 显示错误信息，使人们能够立即明白是什么问题。由于 Option Explicit 语句有助于抓住这种类型的错误，所以最好在所有代码中都使用它。

例如下面一段程序中的开头，就使用了 Option　Explicit 语句检查拼写错误。

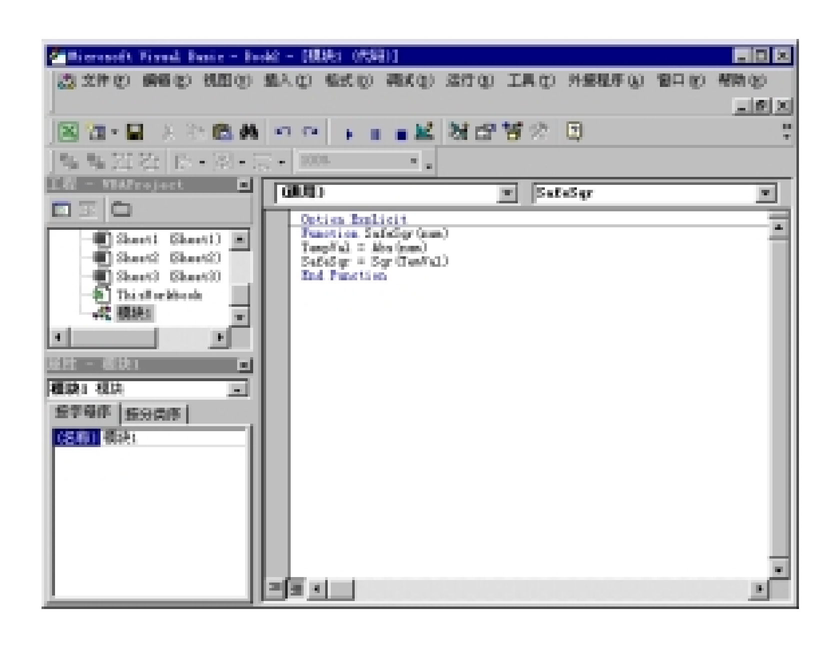


图 4-1　使用 Option Explict 语句的例子

4.3.2　关于变量的作用域和生存期

在 VBA 中，项目包含多个模块，模块又包括多个过程。所谓变量的作用域（Scope），是指变量起作用的区域。一般来说，变量的作用域有三个等级，它们的名称、关键字、使用范围列在表 4-2 中。

表 4-2　变量的作用等级

等级	关键字	使用范围
项目级变量（Global　Level）	Public	适用于所有模块
模块级变量（Module　Level）	Dim, Private	只适用于本模块的所有过程
过程级变量（Local　Level）	Dim, Static	只在本过程中有效

下面举例说明它们各自的声明格式。

Public　GlobalX　As　Integer

'项目级变量，即全局变量 GlobalX 为整型。

Private | Dim　ModY　As　String

'模块级变量 ModY 为字符串类型。

Static | Dim　LocZ　AS　 Date

'变量 LocZ 为过程级的日期型变量。

下面将分别介绍以上三种等级变量的生存期（Lifetime），即变量所占内存空间在何时分配（ Allocate ），何时释放（ Free ）。

对于用 Public 声明的全局变量，因为它可以在所有程序或者函数中使用，其生存期随着程序的执行而产生，程序的结束而结束。

对于用 Private 或者 Dim 声明的模块级变量，将伴随着模块的产生而存在，模块的释放而消失。

对于用 Static 或者 Dim 声明的局部变量，其生存期伴随程序或者函数的执行而生成，程序或者函数的结束而消失。因为局部变量具有这样的特点，就不应该把要保存的数据放在局部变量中，否则，数据将随程序或函数的结束而丢失。

注意：全局变量应该在模块顶部的声明段中加以声明，如图 4-3 所示。而模块变量在模块顶部的声明段中声明，如图 4-4 所示。局部变量在过程顶部声明即可，如图 4-5 所示。

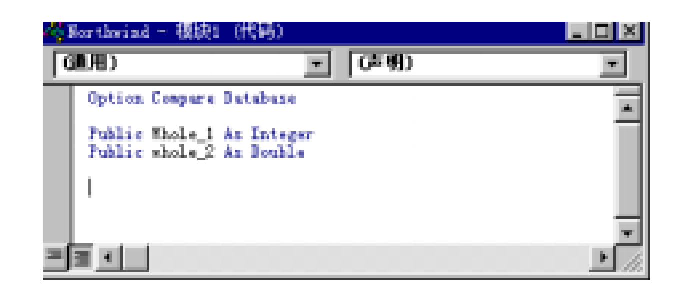


图 4-2　全局变量定义

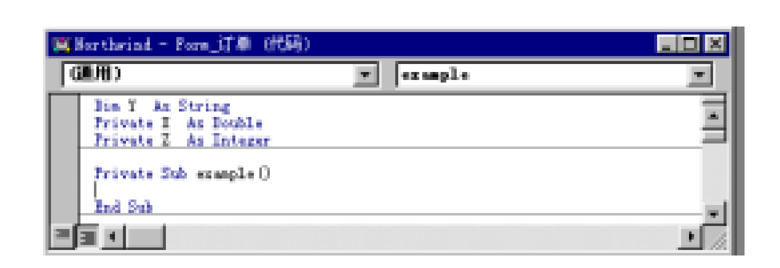


图 4-3　模块级变量的定义



图 4-4　局部变量的定义

4.3.3　数组变量

如果有过其它语言的编程经历，那么对数组的概念应该比较熟悉。用一个名称代表一系列变量，并用索引值（Index）引用各个变量，就是数组。由于有了数组，可以缩短和简化程序，因此可以用索引值设计一个循环，来高效的处理多种情况。

在 Visual Basic 中，有两种类型的数组：固定大小的数组和动态数组。固定大小的数组总是保持同样的大小，而动态数组在运行时大小可以改变。

下面介绍固定大小的数组和动态数组的创建与性质，并附加讲一下有关多维数组的知识。

1. 固定大小的数组

声明固定大小的数组有三种方法，使用哪一种方法取决于数组的有效范围。

（1）创建公用数组，在模块声明段中用 Public 语句声明数组。

（2）创建模块级数组，在模块声明段中用 Private 语句声明数组。

（3）创建局部数组，在过程中用 Private 语句声明数组。

一般的格式是：

Dim 数组名称 （[索引下界 To] 索引上界 ）[As 类型]

例如：

Dim Count(1 To 16) As Integer

'声明了具有 16 个元素的整型数组，索引号是从 1 到 16，即变量 Count（1），Count（2），'Count（3）、……，Count（16）。当缺省索引下界时，默认为 0。

例如：

Dim Students(7) As Long

'声明一个具有 8 个元素的长整数型数组，索引号从 0 到 7，各个变量是 Student（0），'Student（1），Student（2），……，Student（7）。

为建立公用数组，直接用 Public 取代 Dim：

```
Public Sums(100 To 200) As Double
```

还有可能建立 Variant 数据类型的数组，并与不同数据类型的数组共处一处。举下面的例子：

```
Private Sub Command1_Click()
    Dim int1 As Integer
    '声明计数器变量。
    Dim count1(5) As Integer
    '声明并放置整数数组。
        For int1=0 To 4
            count1(int1)=5
        Next int1
    Dim count2(5) As String
    '声明并放置字符串数组。
        For int1=0 To 4
            count2(int1)="World"
        Next int1
    Dim arr(2) As Variant
```

```
'声明拥有两个元素的新数组。
    arr(1)=count1()
    arr(2)=count2()
    '将其它数组移居到新数组。
    MsgBox  arr(1)(2)
    MsgBox  arr(2)(3)
    '显示每一个数组成员。
End  Sub
```

这些代码建立了两个数组，一个包含整数；另一个包含字符串。然后又声明了第三个 Variant 类型的数组，将整数和字符串数组放置于其中。

2. 动态数组

数组到底要有多大，有时可能不清楚。所以希望在运行时能够改变数组大小的。动态数组就可以在任何时候改变大小。在 Visual Basic 中，动态数组最灵活、最方便，而且有助于系统管理内存。例如，动态数组可以在短时间内使用一个大数组，在不使用时，自动将内存空间释放。

创建动态数组的方法是：

（1）Public 语句定义公用数组，用 Dim 语句定义模块级的数组，或者在过程中用 Static 或 Dim 语句定义局部数组。然后，给这些数

组附加一个空维数表，就声明了一个动态数组。即：

Dim　DynArray()

（2）ReDim 语句分配实际元素数目。即：

ReDim　DynArray(N+1)

ReDim 语句实际上是可执行语句，在应用程序中运行时应该执行一个操作。这一点与 Static 语句和 Dim 语句不同。另外，ReDim 语句只能出现在过程中。每个 ReDim 语句，对于每一维的数值，都能改变元素的数目和上下界。注意，数组的维数不可变。

具体的例子：

```
Dim　Array1()　As　Integer
'下面在过程中分配空间。
Sub　CalcValuesNow()
        …
        …
        …
    ReDim　Array1(14,16)
    'ReDim　语句给 Array 分配了一个 15×17 的整数矩阵。
End　Sub
```

此外还有一个方法可以设置动态数组的边界，用变量设置动态数组的边界：

ReDim　Array1 (X,Y)

改变 X 或 Y 的值，就可以改变数组的上下界。

每次执行 ReDim 语句时，当前存储的数据就会全部丢失，因为它是在过程中声明的函数。有时希望数组大小可变却不丢失数据。这时使用具有 Preserve 关键字的 ReDim 语句就可以做到这一点。例如：

ReDim　Preserve　DynArray(Ubound(DynArray)+1)

'使用 Ubound 函数引用上界，使数组扩大，增加一个元素，而数据并未丢失。

在使用 Preserve 关键字时，只能改变最后一维的上界，如果改变了其它维数的上下界，或者改变了最后一维的下界，那么，运行时就会出错。所以应该这样写程序：

ReDim　Preserve　Array(10，Ubound(Array，2)+1)

而不能这样编程：

ReDim　Preserve　Array(Ubound(Array，1)+1，10)

3.　多维数组

有时需要记录数组中的相关信息，这时应该使用多维数组储存。可以用下面的语句声明一个过程内的 9×9 的二维数组。

Static　Matrix　(8,8)　As　Double

可以用显式下界声明两个维数或两个维数中的任意一个：

Static　Matrix　(1 To 10,1 To 10)　As　Double

再将这些推广到二维以上的数组：

Dim　ThreeD　(3，1　To　9，1　To 17)

这就建立了三维数组，大小是 4×9×17。这个三维数组的元素总数是它们三个维数的乘积，即 612。

4.3.4 静态变量

变量除了作用域的概念之外，还有存活期（Persistence）的概念，变量在这一期间能够保存它的值。通常，当一个过程执行完毕，它的过程级别的变量就应该不存在。但是，VBA 用 Static 声明过程级别的静态变量，从而在过程结束后仍然保留变量的值。

在过程内部声明静态变量，其用法与 Dim 语句基本一样。

如：Static　Width

下面举出一个简单的应用示例。

```
Function　SoldTotal(num)
    Static　CakeSold
    CakeSold=CakeSold+num
    SoldTotal=CakeSold
End　Function
```

此函数将从前的销售值与一个新收入相加，来计算新的销售总值。如果不用 Static，而用 Dim 声明 SoldTotal，那么以前的累计值将不

会保留下来，函数只好返回调用它的那个值。

为使过程中所用的局部变量都成为静态变量，可以在过程开头加上关键字 Static：

Static Function SoldTotal(num)

此时，无论它们是用 Static、Dim、隐式声明还是 Private 声明，过程的所有变量均变成静态变量。

4.4 语句和控制结构

本节主要介绍与控制过程有关的表选择的 If、Select Case 语句、表循环的 Do…Loop、While…Wend 语句、For…Next 语句、Go…To 语句以及 Exit 语句等等。

至于像 Beep 语句可以通过计算机喇叭发出一个声调，或者 Call 语句可以调用需要参数的过程，ChDrive 语句用来改变当前的驱动器等内容，在此就不必细细讨论了。如果在将来的应用中需要，可以通过联机帮助中的 Visual Basic 语言参考的语句部分。在这些语句之中，有如何为变量和传给过程的参数设置缺省的数据类型的语句，有在过程级别中用于为动态数组变量重新分配存储空间的语句，有用来重新初始化大小固定的数组的元素以及释放动态数组存储空间的语句，有用来模拟错误的发生的语句，有将一个已打开的磁盘文件读入一个变量之中的语句，还有用来初始化随机数生成器的语句等等。在编程中

熟练应用这些语句，可以达到事半功倍的效果，并能使你的程序更多姿多彩。

下面，开始详细介绍这些过程控制语句，并在本节的结束部分大致介绍一下 VBA 中具有其它作用的某些重要语句。

4.4.1 If Then Else 语句

最简单的条件语句，作用是根据表达式的值，有条件地执行一组语句。

一般写成单行形式：

```
If condition Then [statements][Else elsestatements]
```

或者，可以使用块形式的语法：

```
If condition Then
[statements]
[ElseIf condition-n Then
[elseifstatements] ...
[Else
[elsestatements]]
End If
```

可以使用单行形式（第一种语法）来做简单的测试。但是，块形式（第二种语法）则提供了更强的结构化与适应性，并且通常也是比

较容易阅读、维护及调试的。

If...Then...Else 语句的语法具有以下几个部分，如表 4-3 所示。

表 4-3　If…Else…Then 语句的组成部分

部分	描述
Condition	一个或多个具有下面两种类型的表达式： 数值表达式或字符串表达式。 其运算结果为 True 或 False。如果 condition 为Null，则 condition 会视为 False
Statements	在块形式中是可选参数；但是在单行形式中，且没有 Else 子句时，则为必要参数。一条或多条以冒号分开的语句，它们在 condition 为 True 时执行
Condition-n	与 condition 同
elseifstatements	一条或多条语句,它们在相关的 condition-n 为 True 时执行
elsestatements	一条或多条语句，它们在前面的 condition 或 condition-n 都不为 True 时执行。

✍ 注意 在单行形式中，按照 If...Then 判断的结果也可以执行多条语句。所有语句必须在同一行上并且以冒号分开，如下面语句所示：

If A > 10 Then A = A + 1:　B = B + A ：　C = C + B

在块形式中，If 语句必须是第一行语句。其中的 Else、ElseIf，和 End If 部分可以只在之前加上行号或行标签。If 块必须以一个 End If 语句结束。

要决定某个语句是否为一个 If 块,可检查 Then 关键字之后是什么。如果在 Then 同一行之后，还有其它非注释的内容，则此语句就是单

行形式的 If 语句。

Else 和 ElseIf 子句都是可选的。在 If 块中，可以放置任意多个 ElseIf 子句，但是都必须在 Else 子句之前。If 块也可以是嵌套的。

If　Else　Then 语句的执行原则是：

当程序运行到一个 If 块（第二种语法）时，condition 将被测试。如果 condition 为 True，则在 Then 后的语句会被执行。如果 condition 为 False，则每个 ElseIf 部分的条件式（如果有的话）会依次计算并加以测试。如果找到某个为 True 的条件时，则其紧接在相关的 Then 之后的语句会被执行。如果没有一个 ElseIf 条件式为 True，则程序会执行 Else 部分的语句。而在执行完 Then 或 Else 之后的语句后，会从 End If 之后的语句继续执行。

✍ 提示：根据单一表达式来执行多种可能的动作时，Select Case 语句更为有用。不过，TypeOf objectname Is objecttype 子句不能在 Select Case 语句中使用。而可以在 If　Else Then 语句中使用。注意 TypeOf 不能与诸如 Long、Integer 以及其它不是 Object 的固定数据类型一起使用。

4.4.2　Select Case 语句

根据表达式的值，来决定执行几组语句中的其中之一。

如果在 If　Else　Then 语句中，判断条件都是相同的表达式与不同

的数值进行比较时，则可使用 Select Case 语句。

语法：

Select Case　表达式

[Case　表达式列表 1

[语句陈述 1]]

[Case　表达式列表 2

[语句陈述 2]]…

…

[Case Else

[语句陈述 n]]

End Select

Select Case 语句的语法具有以下几个部分，如表 4-4 所示。

表 4-4　Select Case 语句的语法描述

部分	描述
表达式	任何数值表达式或字符串表达式
表达式列表	其形式为表达式 1，表达式 2，表达式 1 To 表达式 n，Is 比较运算符的数值表达式的一个或多个组成的分界列表。To 关键字可用来指定一个数值范围。如果使用 To 关键字，则较小的数值要出现在 To 之前。使用 Is 关键字时，则可以配合比较运算符（除 Is 和 Like 之外）来指定一个数值范围。如果没有提供，则 Is 关键字会被自动插入
语句陈述	一条或多条语句，当表达式匹配表达式列表中的任何部分时执行

语句陈述 n	一条或多条语句，当表达式不匹配 Case 子句的任何部分时执行

如果表达式匹配某个 Case 子句的表达式，则在 Case 子句之后，直到下一个 Case 子句的语句陈述会被执行；如果是最后一个子句，则会执行到 End Select。然后控制权会转移到 End Select 之后的语句。如果表达式匹配一个以上的 Case 子句中的子句的表达式，则只有第一个匹配后面的语句会被执行。

Case Else 子句用于指明最后一个语句陈述，当表达式和所有的 Case 子句中的表达式都不匹配时，则会执行这些语句。虽然不是必要的，但是在 Select Case 区块中，最好还是加上 Case Else 语句来处理不可预见的表达式的值。如果没有 Case 子句的表达式列表与之相匹配，而且也没有 Case Else 语句，则程序会从 End Select 之后的语句继续执行。

可以在每个 Case 子句中使用多重表达式或使用范围。

例如，下面的语句是正确的：

Case 1 To 4, 7 To 9, 11, 13, Is > MaxNumber

✍ 注意：Is 比较运算符和使用在 Select Case 语句中的 Is 关键字并不相同。

也可以针对字符串指定范围和多重表达式。在下面的例子中，Case 所匹配的字符串为：等于 everything、按英文字母顺序落入从 nuts 到

soup 之间的字符串、以及 TestItem 所代表的当前值。

Case "everything", "nuts" To "soup", TestItem

Select Case 语句也可以是嵌套的。但每个嵌套的 Select Case 语句必须要有相应的 End Select 语句。

下面举出几个 Select Case 语句和 If Else Then 语句的简单例子，对比说明，加深理解。

例 1：假定要实现在"编辑"菜单中添加命令的操作。先用 Select Case 语句来写函数。

```
Private Sub menu_Click(Index As Integer)
    Select Case Index
      Case 0
      '实现剪切命令。
            CopyActiveControl
            '首先调用通用过程。
            ClearActiveControl
      Case1
      '实现复制命令。
            CopyActiveControl
      Case2
      '实现清除命令。
```

```
        ClearActiveControl
    Case3
    '实现粘贴命令。
        PasteActiveControl
    Case Else
    '实现显示找到对话框的命令。
        FrmFind.Show
  End Select
End Sub
```

例 2：同样的命令，用 If Else Then 语句来完成。

```
Private Sub menu_Click(Index As Integer)
If   Index=0 Then
        CopyActiveControl
        '首先调用通用过程。
        ClearActiveControl
        '实现剪切命令。
    ElseIf   Index=1 Then
        CopyActiveControl
        '实现复制命令。
    ElseIf   Index=2 Then
```

```
        ClearActiveControl
        '实现复制命令。
    ElseIf  Index=3 Then
        PasteActiveControl
        '实现粘贴命令。
    Else
        FrmFind.Show
        '实现显示找到对话框的命令。
End If
```

上述两种语句，即 Select Case 和 If Else Then 语句，都是 Visual Basic 支持的过程判定结构。它们能够根据条件测试的结果执行不同的操作。下面介绍的 Do Loop 语句是循环结构语句。

4.4.3 Do loop 语句

当条件为 True 时，或直到条件变为 True 时，Do Loop 语句重复执行一个语句块中的命令，且重复次数不定。

格式：Do [{While | Until} condition]

```
        [statements]
        [Exit Do]
        [statements]
```

Loop

或者可以使用下面这种语法：

Do

[statements]

[Exit Do]

[statements]

Loop [{While | Until} condition]

Do Loop 语句的语法具有以下几个部分：

Condition（条件）是数值表达式或字符串表达式，其值为 True 或 False。如果 condition 是 Null，则 condition 会被当作 False。

Statement（语句陈述）是一条或多条命令，它们将被重复当或直到 condition 为 True。

两种表达方式的不同之处是：

如果使用

Do Until Condition

Statement

Loop 时，要末循环 0 次，要末循环多次；而使用

Do

Statement

Loop Until condition 时，至少要循环一次。

Do Loop 语句执行的原则是：首先判断条件，如果为 True（非零），就执行语句的陈述，然后再回去判断条件；如果条件为 False（或零），立即跳出该语句。

下面举一个例子说明语法的具体部分以及执行原则，以便加深理解。

例 3：以下过程是计算某一特定字符串在另一字符串中出现的次数，只要发现目标就执行循环。

```
Function CountStr (somestr,target)
    Dim position,count
    Position=1
    Do While FindStr(position,somestr,target)
        Position= FindStr(position,somestr,target)+1
        Count=count+1
```

'如果目标字符串未出现在另一字符中，则 FindStr 函数返回 0，就不再执行循环。

```
    Loop
    CountStr=count
End Function
```

在 Do...Loop 中可以在任何位置放置任意个 Exit Do 语句，随时跳出 Do...Loop 循环。Exit Do 通常用于条件判断之后，例如 If...Then，

在这种情况下，Exit Do 语句将控制权转移到紧接在 Loop 命令之后的语句。

如果 Exit Do 使用在嵌套的 Do...Loop 语句中，则 Exit Do 会将控制权转移到 Exit Do 所在位置的外层循环。

例 4：本示例示范如何使用 Do...Loop 语句。内层的 Do...Loop 语句循环到第 10 次时将标志值设置为 False，并用 Exit Do 语句强制退出内层循环。外层循环则在检查到标志值为 False 时，马上退出。

```
Dim Check,Counter
Check=True: Counter = 0
'设置变量初始值。
Do
'外层循环。
    Do While Counter < 20
   '内层循环。
        Counter = Counter + 1
      '计数器加一。
        If Counter = 10 Then
      '如果条件成立。
            Check = False
         '将标志值设成 False。
```

```
            Exit Do
          '退出内层循环。
        End If
    Loop
Loop Until Check = False
'退出外层循环。
```

例 5：在下面的 ChkFirstWhile 过程中，在进入循环之前检查条件。如果将 myNum 的值由 20 替换成 9，则循环中的语句将永远不会运行。在 ChkLastWhile 过程中，在条件变成 False 之前循环中的语句只执行一次。

```
Sub ChkFirstWhile()
    counter = 0
    myNum = 20
    Do While myNum > 10
        myNum = myNum - 1
        counter = counter + 1
    Loop
    MsgBox "The loop made " & counter & " repetitions."
End Sub
```

```
Sub ChkLastWhile()
    counter = 0
    myNum = 9
    Do
        myNum = myNum - 1
        counter = counter + 1
    Loop While myNum > 10
    MsgBox "The loop made " & counter & " repetitions."
End Sub
```

例 6：在下面的示例中，myNum 被赋予一个会造成无穷循环的值。而 If...Then...Else 语句会去检查这个情况然后退出，以避免无穷循环。

```
Sub ExitExample()
    counter = 0
    myNum = 9
    Do Until myNum = 10
        myNum = myNum - 1
        counter = counter + 1
        If myNum < 10 Then Exit Do
    Loop
    MsgBox "The loop made " & counter & " repetitions."
```

End Sub

✍ 注意：可以按 Esc 或 Ctrl+Break 键来终止无穷循环。

4.4.4 For Next 语句

For Next 语句以指定次数来重复执行一组语句。

在事先不知道循环需要执行多少次时，应该使用 Do 循环，而事先知道要执行多少次时，最好使用 For Next 语句。它与 Do 循环不同，在循环过程中使用计数器变量，所以每循环一次，计数器变量就自动增加。

格式：

```
For counter = start To end [Stepincrement]
    [statements]
    [Exit For]
    [statements]
Next [counter]
```

For…Next 语句的语法具有 counter，start，end，step 和 statement 几个部分，具体意义如表 4-5 所示。

表 4-5 For…Next 循环语句的参数意义

部分	描述
Counter	用做循环计数器的数值变量。这个变量是数值类型的。
Start	counter 的初值。

End	counter 的终值。
Step	counter 的步长。如果没有指定，则 step 的缺省值为 1。
Statements	放在 For 和 Next 之间的一条或多条语句，它们将被执行指定的次数。

其中，step 参数可以是正数或负数。step 参数值决定循环的执行情况。

如果 start 小于或等于 end，step 的值为正数或 0；如果 start 大于 end，step 的值为负数。

关于 For…Next 语句的执行原则是：

（1）在执行 For 循环时，设置 counter 的值等于 start。

（2）测试 counter 是否大于 end。在 step 非负的情况下，若 counter 大于 end，则此时退出循环（对于 step 是负数的情况，则检验 counter 是否小于 end）。

（3）执行语句陈述的内容。

（4）Counter 增加 1，或者增加 step。

（5）重复步骤 2 到 4。

例 7：用以下代码打印出所有的有效的屏幕字体名称。

```
Private  Sub  Writing_Click()
    Dim  I  as  Integer
    For  I=0 To Scree.FontCount
    '从第 0 号字体一直循环到所有字体数目的最后一个。总的字体
```

数目是已知的。

```
        Print    Scree.Fonts(i)
        '再将它们一一打印出来。
    Next
End Sub
```

像这种已知循环次数的操作，使用 For…Next 语句会比盲目循环要节省空间，还可以减少错误。

当所有循环中的语句都执行后，step 的值会加到 counter 中。此时，循环中的语句可能会再次执行（基于循环开始执行时同样的测试），也可能是退出循环并从 Next 语句之后的语句继续执行。

> ✍ *在循环中改变 counter 的值，将会使程序代码的阅读和调试变得更加困难。*

循环中可以在任何位置放置任意个 Exit For 语句，随时退出循环。

可以将一个 For...Next 循环放置在另一个 For...Next 循环中，组成嵌套循环。不过在每个循环中的 counter 要使用不同的变量名。下面的体系结构是正确的：

```
For I = 1 To 10
   For J = 1 To 10
      For K = 1 To 10
……
```

```
        Next K
    Next J
Next I
```

✍ 注意：如果省略 Next 语句中的 counter，就像 counter 存在时一样执行。但如果 Next 语句在它相对应的 For 语句之前出现，则会产生错误。

例 8：使用上述结构给某个二维数组（10×10）赋值为奇数的整数值，即 1，3，5，7，9……。

```
Dim  Arr(10,10)  As  Integer
Dim  I  As  Integer
Dim  J  As  Integer
Dim  K  As  Integer
K=1
For  I=1 To 10
   For  J=1 To 10
'利用嵌套的 For  Next 语句构成循环结构，进行赋值。
        Arr(I,J)=K
        K=K+2
'从 1 开始的奇数顺序赋给数组的元素。
   Next  J
```

Next I

4.4.5 For Each Next 语句

For Each Next 语句针对一个数组或集合中的每个元素，重复执行一组语句。它与 For Next 语句类似，但它是对数组或对象集合中的每一个元素一组语句，而不是重复语句一定的次数。如果不知道一个集合有多少元素，那么 For…Each…Next 语句非常有用。

语法：For Each element In group

[statements]

[Exit For]

[statements]

Next [element]

For...Each...Next 语句的语法中的参数及其意义如表 4-6 所示。

表 4-6 For…Each..Next 语句的参数及意义

部分	描述
Element	用来遍历集合或数组中所有元素的变量。对于集合来说，element 可能是一个 Variant 变量、一个通用对象变量或任何特殊对象变量。对于数组而言，element 只能是一个 Variant 变量。
Group	对象集合或数组的名称（用户定义类型的数组除外）。
Statements	针对 group 中的每一项执行的一条或多条语句。

如果集合中至少有一个元素，就会进入 For...Each 块执行。一旦进

入循环，便先针对 group 中第一个元素执行循环中的所有语句。如果 group 中还有其它的元素，则会针对它们执行循环中的语句，当 group 中的所有元素都执行完了，便会退出循环，然后从 Next 语句之后的语句继续执行。

在循环中可以在任何位置放置任意个 Exit For 语句，随时可以退出循环。Exit For 经常在条件判断之后使用，例如 If...Then，并将控制权转移到紧接在 Next 之后的语句。

可以将一个 For...Each...Next 循环放在另一个之中来组成嵌套式 For...Each...Next 循环。但是每个循环的 element 必须是唯一的。

> 注意：如果省略 Next 语句中的 element，就像 element 存在时一样执行。如果 Next 语句在它相对应的 For 语句之前出现，则会产生错误。

不能在 For Each...Next 语句中使用用户自定义类型数组，因为 Variant 不能包含用户自定义类型。

这里再次说明使用 For Each…Next 语句的几个限制：

（1）对于集合，element 只能是 Variant 变量，或者一般的 Object 变量，或者“对象浏览器”中列出的对象。

（2）对于数组，element 只能是 Variant 变量。

（3）For..Each..Next 语句不能与用户自定义类型的数组一起使用，因为，Variant 变量不可能包含用户自定义类型。

例 9：用 For Each…Next 语句把一个表的名字加到列表框中。

```
Sub   ListTab()
      Dim   Aobject   As   Database
      '定义一个对象变量。
      Set   Aobject=OpenDatabase("c:\Visual Basic\biao.mdb",True,False)
      '打开一个名称为 biao.mdb 的数据库。
      For Each Tab In Aobject.Tab()
         List1.AddItem   Tab.Name
         '把文件中的每一个表的名字加入到列表框 List1 中。
      Next Tab
End Sub
```

4.4.6 While Wend 语句

只要指定的条件为 True，While Wend 语句会重复执行一系列的语句。

语法：

```
While   condition
[statements]
Wend
```

While...Wend 语句的语法具有以下几个部分：

表 4-7 While…Wend 语句中参数的具体意义

部分	描述
Condition	数值表达式或字符串表达式，其计算结果为 True 或 False。如果 condition 为Null，则 condition 会视为 False。
Statements	一条或多条语句，当条件为 True 时执行。

如果 condition 为 True，则所有的 statements 都会执行，一直执行到 Wend 语句。然后再回到 While 语句，并再一次检查 condition，如果 condition 还是为 True，则重复执行。如果不为 True，则程序会从 Wend 语句之后的语句继续执行。

While...Wend 循环也可以是多层的嵌套结构。每个 Wend 匹配最近的 While 语句。

事实上，Do...Loop 语句提供了一种结构化与适应性更强的方法来执行循环。

例 10：本示例使用 While...Wend 语句来增加计数变量的值。如果条件判断值为 True，则循环内的语句将一直执行下去。

Dim Counter

Counter = 0

'设置变量初值。

```
While   Counter<20
'测试计数器的值。
    Counter = Counter + 1
   '将计数器的值加一。
Wend

'当 Counter>19 时则循环终止。
Debug.Print Counter
'在“立即”窗口中显示数字 20。
```

4.4.7 Exit 语句

Exit 语句是退出 Do...Loop、For...Next、Function、Sub 或 Property 语句、函数或过程的代码块。

Exit Do 提供一种退出 Do...Loop 循环的方法；Exit For 提供一种退出 For 循环的方法；Exit Function 立即从包含该语句的 Function 过程中退出。程序会从调用 Function 的语句之后的语句继续执行；Exit Property 立即从包含该语句的 Property 过程中退出，程序会从调用 Property 过程的语句之后的语句继续执行；Exit Sub 立即从包含该语句的 Sub 过程中退出，程序会从调用 Sub 过程的语句之后的语句继续

执行。

例 11：本示例使用 Exit 语句退出 For...Next 循环、Do...Loop 循环及子过程。

```
Sub ExitStatementDemo()
Dim I,MyNum
    Do
'建立无穷循环。
            For I = 1 To 1000
        '循环 1000 次。
                MyNum = Int(Rnd * 1000)
            '生成一随机数码。
                Select Case MyNum
            '检查随机数码。
                    Case 7: Exit For
                '如果是 7，退出 For...Next 循环。
                    Case 29: Exit Do
                '如果是 29，退出 Do...Loop 循环。
                    Case 54: Exit Sub
                '如果是 54，退出子过程。
                End Select
```

Next I

Loop

End Sub

4.4.8 Go To 语句

可以无条件地转移到过程中指定的行。

语法：GoTo line

与 GoSub line Return 中的 line 参数一样，可以是任意的行标签或行号。

GoTo 只能跳到它所在过程中的行。

✍ 太多的 GoTo 语句，会使程序代码不容易阅读及调试。建议尽可能使用结构化控制语句(Do...Loop、For...Next、If...Then...Else、Select Case)。

例 12：本示例使用 GoTo 语句在一个过程内的不同程序段间作流程控制，不同程序段用不同的“程序标签”来区隔。

Sub GotoStatementDemo()

Dim Number, MyString

Number = 1

' 设置变量初始值。

' 判断 Number 的值以决定要完成那一个程序区段（以“程序标

签”来表式）。

```
        If Number = 1 Then GoTo Line1 Else GoTo Line2


    Line1:
        MyString = "Number equals 1"
        GoTo LastLine
      ' 完成最后一行。
    Line2:
        ' 下列的语句根本不会被完成。
        MyString = "Number equals 2"
    LastLine:
        Debug.Print MyString
      ' 将“"Number equals 1"”显示在“立即”窗口。
    End Sub
```

> 注意：由于 GoTo 语句可以任意转向另一语句，所以它相当灵活，但是，在发生错误时，难于调试，应该尽量避免使用它。

4.4.9 关于控制结构

可以通过把控制结构放入另一个控制结构中，形成嵌套的控制结

构。这一点在 For Next 语句中已有例子说明。

在 Visual Basic 中，控制结构的嵌套层数是没有限制的。为了使程序更具有可读性，一般是用缩进对称的书写方式来判定结构或者循环的正文部分。

> ✍ 注意：在第一个 Next 关闭了内存的 For 循环，而最后一个 For 关闭了外层的 For 循环。同样，在嵌套结构的 If 语句中，EndIf 语句自动与最靠近的 If 构成配对。在 Do..Loop 语句的嵌套中，其工作方式基本一样，最内圈的 Loop 语句与最内圈的 Do 语句配套。

关于退出控制结构，用 Exit 语句可以退出 For 循环、Do 循环、子过程或者函数过程。尤其是 Exit For 和 Exit Do，非常有用，它们可以立即退出循环。

例 13：利用 Exit For 语句，实现立即退出 For　Next 语句循环。本例是找到打印机和屏幕的共有字体，并将它们打印出来。

```
Private Sub Mod_Click()
    Dim  SFont,PFont
    For  Each  Sfont  In  Scree.Fonts()
      For  Each  PFont  In  Printer.Fonts()
        If  Sfont=Pfont Then
            Print Sfont
```

```
            Exit    For
        End If
      Next    Pfont
   Next    Sfont
End Sub
```

也可以从控制结构内部退出一个过程。利用 Exit Sub 和 Exit Function 的语句，可以立即退出子过程，而且出现的次数可以随需要而定。

例 14：现在要改动例 13，使得在找到打印机和屏幕的共同字体后，只打印其中的一种，使用 Exit Sub 语句。

```
Private Sub Mod_Click()
   Dim    SFont,PFont
   For    Each    Sfont    In    Scree.Fonts()
      For    Each    PFont    In    Printer.Fonts()
        If    Sfont=Pfont Then
            Print Sfont
            Exit    Sub
        End If
      Next    Pfont
   Next    Sfont
```

End Sub

4.4.10　其它语句简介

AppActivate 语句激活一应用程序窗口。

Beep 语句通过计算机喇叭发出一个声调。

Call 语句将控制权转移到一个 Sub 过程，Function 过程，或动态链接库（DLL）过程。

ChDir 语句改变当前的目录或文件夹。

MkDir 语句创建一个新的目录或文件夹。

RmDir 语句删除一个存在的目录或文件夹。

ChDrive 语句改变当前的驱动器。

Close 语句关闭 Open 语句所打开的输入/输出(I/O)文件。

Open 语句能够对文件输入/输出(I/O)。

Reset 语句关闭所有用 Open 语句打开的磁盘文件。

Width # 语句将一个输出行的宽度指定给用 Open 语句打开的文件。

Const 语句声明用于代替文字量的常数。

Date 语句设置当前系统日期。

Time 语句设置系统时间。

Declare 语句用于在模块级别中声明对动态链接库(DLL)中外部过

程的引用。

DefType 语句在模块级别上，为变量和传给过程的参数，设置缺省数据类型，以及为其名称以指定的字符开头的 Function 和 Property Get 过程，设置返回值类型。

Type 语句在模块级别中使用，用于定义包含一个或多个元素的用户自定义的数据类型。

DeleteSetting 语句在 Windows 注册表中或(Macintosh 中)应用程序初始化文件中的信息，从应用程序项目里删除区域或注册表项设置。

SaveSetting 语句在 Windows 注册表中或(Macintosh 中)应用程序初始化文件中的信息保存或建立应用程序项目。

Dim 语句声明变量并分配存储空间。

ReDim 语句在过程级别中使用，用于为动态数组变量重新分配存储空间。

End 语句结束一个过程或块。

Enum 语句定义枚举类型。

Erase 语句重新初始化大小固定的数组的元素，以及释放动态数组的存储空间。

Error 语句模拟错误的发生。

On　Error 语句启动一个错误处理程序并指定该子程序在一个过程中的位置；也可用来禁止一个错误处理程序。

Event 语句用来定义用户自定义的事件。

RaiseEvent 语句引发在一个类、窗体、或者文档中的模块级中声明的一个事件。

FileCopy 语句复制一个文件。

Function 语句用来声明 Function 过程的名称，参数以及构成其主体的代码。

Sub 语句声明子过程的名称，参数，以及构成其主体的代码。

Get 语句将一个已打开的磁盘文件读入一个变量之中。

GoSub…Return 语句在一个过程中跳到另一个子程序中执行，执行后再返回。

On GoSub 语句根据表达式的值，转到特定行执行。

On GoTo 语句根据表达式的值，转到特定行执行。

Implement 语句指定要在包含该语句的类模块中实现的接口或类。

Input#语句从已打开的顺序文件中读出数据并将数据指定给变量。

Line Input#语句从已打开的顺序文件中读出一行并将它分配给 String 变量。

Kill 语句从磁盘中删除文件。

Let 语句将表达式的值赋给变量或属性。

Load 语句装载一对象但却不显示。

Unload 语句从内存中删除一个对象。

Lock，Unlock 语句对于用 Open 语句打开的全部文件或一部分文件，其它进程所进行的控制访问。

Lset 语句在一字符串变量中将一字符串往左对齐，或是将一用户定义类型变量复制到另一用户自定义类型变量。

RSet 语句在一字符串变量中将一字符串往右对齐。

Mid 语句在一个字符串中以另一个字符串中的字符替换其中指定数量的字符。

Name 语句重新命名一个文件、目录、或文件夹。

Option Base 语句在模块级别中使用，用来声明数组下标的缺省下界。

Option Compare 语句在模块级别中使用，用于声明字符串比较时所用的缺省比较方法。

Option Explicit 语句在模块级别中使用，强制显式声明模块中的所有变量。

Option Private 语句在允许引用跨越多个工程的主机应用程序中使用 Option Private Module，可以防止在模块所属的工程外引用该模块的内容。在不允许这种引用的主机应用程序中，例如，Visual Basic 的独立方式版本，Option Private 就不起作用。

Print#语句将格式化显示的数据写入顺序文件中。

Private 语句在模块级别中使用，用于声明私有变量及分配存储空

间。

Static 语句在过程级别中使用，用于声明变量并分配存储空间。在整个代码运行期间都能保留使用 Static 语句声明的变量的值。

Public 语句在模块级别中使用，用于声明公用变量和分配存储空间。

Property Get 语句声明 Property 过程的名称，参数以及构成其主体的代码，该过程获取一个属性的值。

Property Let 语句声明 Property Let 过程的名称，参数以及构成其主体的代码，该过程给一个属性赋值。

Property Set 语句声明 Property 过程的名称，参数以及构成其主体的代码，该过程设置一个对象引用。

Put 语句将一个变量的数据写入磁盘文件中。

Randomize 语句初始化随机数生成器。

Rem 语句用来在程序中包含注释。

Resume 语句在错误处理程序结束后，恢复原有的运行。

Seek 语句在 Open 语句打开的文件中，设置下一个读/写操作的位置。

SendKeys 语句将一个或多个按键消息发送到活动窗口，就如同在键盘上进行输入一样。

Set 语句将对象引用赋给变量或属性。

SetAttr 语句为一个文件设置属性信息。

Stop 语句表示暂停执行。

With 语句在一个单一对象或一个用户定义类型上执行一系列的语句。

Write#语句将数据写入顺序文件。

第五章　过程与函数

5.1　过程概述

将程序分为若干较小的逻辑部件，这些部件称为过程。过程可以简化程序设计任务，还可以增强和扩展 Visual Basic 的构件。

用过程编写程序有两个优点：

（1）可以把程序划分为离散的单元，每个单元都可以单独调试。

（2）一个过程往往不必更改，或者只需稍加修改，就可以成为另一个程序的构件。

过程还可以用于共享任务或压缩重复任务。例如压缩频繁的计算、压缩文本、控件和数据库的操作。

在 Visual Basic 中一般有以下过程：

Sub 过程，不返回值；

Founction 过程，返回值；

Property 过程，返回值并指定值，还设置对象的引用。

Sub 过程与 Function 过程的相似之处是：它们都是一个可以获取参数，执行一系列语句，以及改变其参数值的独立过程。而与 Function 过程不同的是：带返回值的 Sub 过程不能用于表达式。

5.2　子过程

5.2.1　Sub 过程

Sub 过程又称为子过程。它是在响应事件时执行的代码块。如果将模块中的代码分成子过程后，在应用程序中查找和修改错误代码就容易多了。

Sub 过程是一系列由 Sub 和 End Sub 语句所包含起来的 Visual Basic 语句，它们会执行动作却不能返回一个值。Sub 过程可以有参数，例如常数、变量、或是表达式等来调用它。如果一个 Sub 过程没有参数，则它的 Sub 语句必须包含一个空的圆括号。

具体的子过程的语法是：

```
[Private|Public|Friend][Static]Sub   name  （arglist）
        statements
End  Sub
```

其中的各个参数的意义如表 5-1 所示。

表 5-1　Sub 过程的参数及意义

部分	描述
Public	表示所有模块的所有其他过程都可访问这个 Sub 过程。如果在包含 Option Private 的模块中使用，则这个过程在该工程外是不可使用的。
Private	表示只有在包含其声明的模块中的其他过程可以访问该 Sub 过程。

Friend	只能在类模块中使用。表示该 Sub 过程在整个工程中都是可见的，但对对象实例的控制者是不可见的。
Static	表示在调用之间保留 Sub 过程的局部变量的值。Static 属性对在 Sub 外声明的变量不会产生影响，即使过程中也使用了这些变量。
name	Sub 的名称；遵循标准的变量命名约定。
arglist	代表在调用时要传递给 Sub 过程的参数的变量列表。多个变量则用逗号隔开。
statements	Sub 过程中所执行的任何语句组。

其中的 arglist 参数的语法：

[Optional] [ByVal | ByRef] [ParamArray] varname[( )] [As type] [= defaultvalue]

语法的各个部分参见表 5-2。

表 5-2　arglist 参数描述

部分	描述
Optional	如果使用了该选项，则 arglist 中的后续参数都必须是可选的，而且必须都使用 Optional 关键字声明。如果使用了 ParamArray，则任何参数都不能使用 Optional。
ByVal	表示该参数按值传递。
ByRef	表示该参数按地址传递。ByRef 是 Visual Basic 的缺省选项。
ParamArray	只用于 arglist 的最后一个参数，指明最后这个参数是一个 Variant 元素的 Optional 数组。使用 ParamArray 关键字可以提供任意数目的参数。ParamArray 关键字不能与 ByVal，ByRef，或 Optional 一起使用。
varname	代表参数的变量的名称；遵循标准的变量命名约定。
type	传递给该过程的参数的数据类型，如果没有选择参数 Optional，则可以指定用户定义类型，或对象类型。

defaultvalue	任何常数或常数表达式。只对 Optional 参数合法。如果类型为 Object，则显式的缺省值只能是 Nothing。

例 1：下面是某个 Sub 过程。

```
'声明过程命名为 Infor，该 Sub 过程没有参数。

Sub Infor()

'声明字符串变量命名为 name。

Dim name As String

'指定 InputBox 函数的返回值给 name。

name=InputBox(Prompt:="What is your name?")

    If answer = Empty Then

                '条件 If...Then...Else 语句。

     MsgBox Prompt:="You did not enter a name."

                '调用 MsgBox 函数。

    Else

      MsgBox Prompt:="Your name is " & name

     'MsgBox 函数与 name 变量连接。

    End If

    '结束 If...Then...Else 语句。

End Sub

'结束 Sub 过程。
```

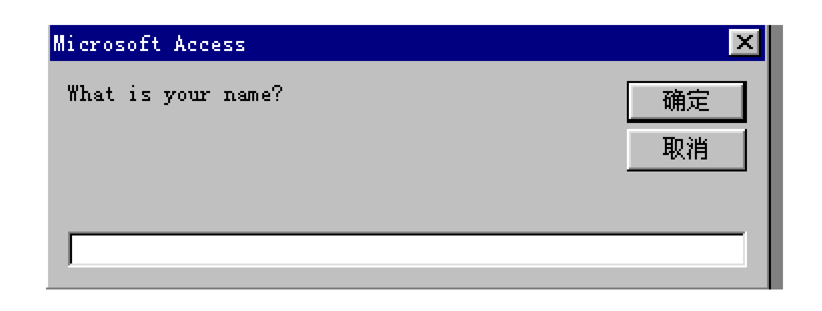


图 5-1　例 1 的运行结果

Exit Sub 语句使执行立即从一个 Sub 过程中退出。程序接着从调用该 Sub 过程的语句的下一条语句执行。在 Sub 过程的任何位置都可以有 Exit Sub 语句。

每次调用子程序名都会执行 Sub 中间的陈述。可以将这些子过程放入标准模块、类模块或者窗体模块中。按照 VBA 中的缺省规定，所有模块中的子过程都是 Public，所以在应用程序中可以随处调用它们。如果没有使用 Static，则在调用之后不会保留局部变量的值。Friend 关键字只能在类模块中使用。不过 Friend 过程可以被工程的任何模块中的过程访问。Friend 过程不会在其父类的类型库中出现，且 Friend 过程不能被后期绑定。

所有的可执行代码都必须属于某个过程。不能在别的 Sub、Function 或 Property 过程中定义 Sub 过程。

例 2：该示例使用 Sub 语句来定义子过程的名称、参数、以及构成子过程主体的代码。

Sub SubComputeArea(Length,TheWidth)

'子过程的定义。子过程带有两个参数。

Dim Area As Double

'声明局部变量。

If Length = 0 Or TheWidth = 0 Then

'如果有一个长或者宽的参数等于 0，

Exit Sub

'就立即退出子过程。

End If

Area = Length * TheWidth

'计算矩形的面积，即长乘以宽。

Debug.Print Area

'将面积显示在调试窗口。

End Sub

'退出 Sub 过程。

✍ Sub 过程可以是递归的；也就是说，该过程可以调用自己来完成某个特定的任务。不过，递归可能会导致堆栈上溢。通常 Static 关键字和递归的 Sub 过程不在一起使用。

在 Sub 过程中使用的变量分为两类：一类是在过程内显式定义的，另一类则不是。在过程内显式定义的变量（使用 Dim 或等效方法）都是局部变量。对于使用了但又没有在过程中显式定义的变量，除非其

在该过程外更高级别的位置有显示地定义，否则也是局部的。

- 过程可以使用没有在过程内显式定义的变量，但只要有任何在模块级别定义的名称与之同名，就会产生名称冲突。如果过程中使用的未定义的变量与别的过程，常数，或变量的名称相同，则认为过程使用的是模块级的名称。显式定义变量就可以避免这类冲突。可以使用 Option Explicit 语句来强制显式定义变量。
- 不能使用 GoSub、GoTo 或 Return 来进入或退出 Sub 过程。

5.2.2 通用过程和事件过程

在 Visual Basic 中，应该区分通用过程和事件过程这两类子过程。

（1）通用过程

通用过程是告诉应用程序如何完成一项指定的任务。通用过程必须由应用程序来调用，而事件过程则通常处于空闲状态。

建立通用过程的原因之一是几个不同的过程也许要执行相同的动作。如果将公共语句放入一个分离的过程（通用过程），并由事件过程来调用它，就不必重复代码，也容易维护程序。

例如，如果有几个不同的按钮（Click）事件，都调用了“按钮管理器”这一个通用过程。Click 事件中的代码调用“按钮管理器”过程，这个通用过程运行自身的代码，然后将控制返回到各个 Click 事件过

程。

（2）事件过程

当 Visual Basic 中的对象对一个事件的发生作出认定，就自动用相应的事件名称调用事件过程,因为名称在对象和代码之间建立了联系，所以说事件过程事实上是附加在窗体和控件上的。

所有事件过程使用相同的语法。

虽然可以自己编写事件过程，但使用 Visual Basic 提供的代码过程会更方便。从“对象框”中选择一个过程，就可以在“代码编辑器”窗口选择一个模板。

例如，当发生窗体或者控件的事件时（如单击、双击等等），则会调用相应的对象的事件处理程序，这就是事件过程。一般来说，它们的名称是由“_”连接对象名称和事件名称构成，例如 Fonts_Click 表示关于字体的按钮过程。

例 3：下例说明在窗体最大化时，如何使用 Resize 事件过程来重画窗体。当用户按下名为“最大化”的命令按钮时，窗体将最大化，并且 Resize 事件被触发。

如果要试验此示例，请将下列事件过程添加到名为“联系人”的窗体中，而且使该窗体含有一名为“最大化”的命令按钮：

```
Private Sub Maximize_Click()
    DoCmd.Maximize
```

```
End Sub
Private Sub Form_Resize()
    Forms!Contacts.Repaint
End Sub
```

例 4：在以下示例中，Current 事件过程检查“中止”选项按钮的状态。如果按钮被选取，示例将“产品名称”字段的背景颜色设置为红色，以指示该产品状态为“中止”。

如果要试验此示例，请将下列事件过程添加到窗体中，并在该窗体中包含一名为“中止”选项和名为“产品名称”的文本框。

```
Private Sub Form_Current()
    If Me!中止 Then
        Me!ProductName.BackColor = 255
    EndIf
End Sub
```

下一个示例将窗体的 Caption 属性设置为“供应商名称”字段。当焦点在记录之间移动时，标题栏将显示当前供应商的名称。

如果要试验此示例，请将下列事件过程添加到窗体中，并使窗体包含一名为“供应商名称”的文本框。

```
Private Sub Form_Current()
    Me.Caption = Me!供应商名称
```

```
End Sub
```

例 5：在以下示例中，Click 事件过程附属于“只读”复选框。这个事件用于过程设置窗体中另一个控件（“Amount”文本框）的 Enabled 和 Locked 属性。当单击复选框时，事件过程会检查复选框是否选中或被清除，然后再相应地将文本框的属性设置为能够编辑或不能够编辑。

如果要试验这个示例，请将下列事件过程添加到含有一名为“只读”复选框和名为“Amount”文本框的窗体中。

```
Private Sub ReadOnly_Click()
  With Me!Amount
    If Me!ReadOnly = True Then
        Enabled = False'如果选中。
        Locked = True'不能编辑。
        Else
          '如果清除。
          Enabled = True
          '可以编辑。
          Locked = False
      End If
    End With
End Sub
```

例 6：下例使用 NotInList 事件将新输入项添加到组合框中。

要试验此示例，在窗体中创建名为“颜色”的组合框。请将组合框的 LimitToList 属性设置为 Yes。为了操作该组合框，将组合框的 RowSourceType 属性设置为 Value List，并且提供一个用分号分隔的值列表作为 RowSource 属性的设置值。例如，可以提供下列值作为此属性的设置：Red，Green，Blue。然后将下列事件过程添加到窗体中。切换到“窗体”视图并且在组合框的文本框部分输入新值。

```
Private Sub Colors_NotInList(NewData As String, _
    Response As Integer)
    Dim ctl As Control
    '返回指向组合框的 Control 对象。
    Set ctl = Me!Colors
    '提示用户验证要添加的新值。
    If MsgBox("Value is not in list. Add it?",Visual BasicOKCancel) = Visual BasicOK Then
    '将 Response 参数设置为指向。
    '正在添加的数据。
    Response = acDataErrAdded
    '将 NewData 参数中的字符串添加到行来源。
    ctl.RowSource = ctl.RowSource & ";" & NewData
```

```
Else
'如果用户选择了“取消”，将不再出现错误消息
'并且撤消所做的更改。
Response = acDataErrContinue
ctl.Undo
End If
End Sub
```

注意：以上示例显示的是将一个新输入项添加到未绑定组合框中。将一个新的输入项添加到绑定组合框中时，实际上是将值添加到基础数据来源的字段中。在大多数情况下，不能简单地在新记录中添加一个字段，这取决于表中的数据结构，而且可能需要添加一个或多个字段来满足数据要求。例如，新记录必须包含组成主键的所有字段的值。如果需要动态地将输入项添加到绑定组合框中，必须提示用户为所有需要的字段输入数据，并保存新记录，然后重新查询组合框以显示新值。

在介绍完通用过程和事件过程后，要提醒一个值得注意的问题，在开始为控件编写事件过程前，最好是预先设置控件的 Name 属性。如果对一个控件附加一个过程后又更改控件的名字，那就必须更改过程的名字；否则控件与过程的名字不符，过程就自动变成通用过程。

5.3 函数过程

Visual Basic 中包含内部的函数，如 Sqr、String、Sin 等。这在第二章的内部函数中已有详细介绍。此外，还可以用 Function 语句编写 Function 过程。一个函数过程包括它的名称，参数以及构成其主体的代码。

语法：[Public | Private | Friend] [Static] Function name [(arglist)] [As type]

```
        [statements]
        [name = expression]
        [Exit Function]
        [statements]
        [name = expression]
    End Function
```

Function 语句的语法包含的部分及意义如表 5-3 所示。

表 5-3 Function 语句的参数及意义

部分	描述
Public	表示所有模块的所有其他过程都可访问这个Function过程。如果是在包含 Option Private 的模块中使用，则这个过程在该工程外是不可使用的。
Private	表示只有包含其声明的模块的其他过程可以访问该 Function 过程。

Friend	只能在类模块中使用。表示该 Function 过程在整个工程中都是可见的，但对于对象实例的控制者是不可见的。
Static	表示在调用之间将保留Function过程的局部变量值。Static属性对在该 Function 外声明的变量不会产生影响，即使过程中也使用了这些变量。
Name	Function 的名称；遵循标准的变量命名约定。
arglist	代表在调用时要传递给 Function 过程的参数变量列表。多个变量应用逗号隔开。
type	Function 过程的返回值的数据类型。
statements	在 Function 过程中执行的任何语句组。
expression	Function 的返回值。

其中的 arglist 参数的语法以及语法各个部分与 Sub 语句的基本相同，在此就不详细列表介绍了。

例 6：下面的例子说明如何给一个名为 BinarySearch 的函数赋返回值。在这个示例中，将 False 赋给了该函数名，表示没有找到某个值。

```
Function BinarySearch(. . .) As Boolean
    If lower > upper    Then
            BinarySearch = False
            Exit Function
          '值未找到，返回一个 False 值。
        End If
End Function
```

例 7：该示例使用 Function 语句返回参数的平方根。

```
'这个用户自定义函数将返回它的参数的平方根。
Function CalculateSquareRoot(NumberArg As Double) As Double
    If NumberArg < 0 Then
      '评估参数。
        Exit Function
      '退出调用过程。
    Else
        CalculateSquareRoot = Sqr(NumberArg)
      '返回平方根。
    End If
End Function
```

例 8：使用 ParamArray 关键字可以使函数接收数目可变的参数。在下面的定义中，FirstArg 是按值传递的。

```
Function CalcSum(ByVal FirstArg As Integer, ParamArray OtherArgs())
Dim ReturnValue
'如果用如下代码调用该函数：
ReturnValue = CalcSum(4,3,2,1)
'则局部变量被赋予以下值：FirstArg = 4，OtherArgs(1) = 3，OtherArgs(2) = 2 等。
```

```
'假设缺省数组下界=1。
'Optional 参数可以带缺省值，可以是除 Variant 之外的任何类型。
'如果函数的参数定义如下：
Function MyFunc(MyStr As String, Optional MyArg1 As
_Integer=5,Optional MyArg2="Dolly")
Dim RetVal
'则可用如下代码调用该函数：
RetVal = MyFunc("Hello",2,"World")
'提供了所有 3 个参数。
RetVal = MyFunc("Test", ,5)
'省略了参数 2。
'参数 1 和参数 3 使用了命名的参数。
RetVal = MyFunc(MyStr:="Hello ",MyArg1:=7)
```

5.4　使用过程

5.4.1　创建和查看过程

在“代码”窗口输入过程头，并按 Enter 键，就创建了一个过程。过程头以 Sub 或 Function 开头，然后接一个名字。

例如：

```
Sub　PriceForm()
```

Function　Font_Click()

要在当前的模块中查看现有的通用过程，可以在“代码”窗口选择“对象框”，再选“通用”，然后在“过程框”中选择过程。要查看事件，与上述步骤相同，只在最后一步更改成在“过程框”中选择事件。

若要查看其他模块的过程，则要按照以下步骤：

（1）“视图”菜单中选取“对象浏览器”。

（2）在“工程/库”中选择工程。

（3）在“类/模块”中选择模块，并在“成员”列表中选择过程。

（4）最后选取“查看定义”。

5.4.2　调用 Sub 和 Function 过程

调用过程有很多技巧，这与过程的类型、位置。在程序中的使用方式有关。

下面分别说明调用 Sub 过程与 Function 过程的方法与注意事项。

调用 Sub 过程有两种方法，下面举例说明。例如要调用名为 MyUse 的 Sub 过程，可以写成：

（1）Call MyUse(First Number,Second Number)

（2）MyUse First Number,Second Number

从其他过程调用一个 Sub 过程时，必须键入过程名称以及任何需

要的参数值。与 Function 过程一样，Sub 过程可以修改传递给它的任何变量的值。但是，调用 Sub 过程不能用它的名字，而且，它不会用名字返回一个值。调用 Sub 过程是一个独立的语句，

- 当使用 Call 语句时，任何参数必须以括号括起来；如省略 Call 关键字，也必须省略括号。

可以使用 Sub 过程去组织其他的过程，因此可以较容易了解并调试它们。

例 10：在下面的示例中，Sub 过程 Main 传递参数值 56 去调用 Sub 过程 MultiBeep。运行 MultiBeep 后，控件返回 Main，然后 Main 调用 Sub 过程 Message。Message 显示一个信息框；当按“确定”键时，控件会返回 Main，接着 Main 退出执行。

```
Sub Main()
    MultiBeep 56
    Message
End Sub
Sub MultiBeep(numbeeps)
    For counter = 1 To numbeeps
        Beep
    Next counter
End Sub
```

```
Sub Message()
    MsgBox "Time to take a break!"
End Sub
```

例 11：下面的示例展示了调用具有多个参数的 Sub 过程的两种不同方法。当第二次调用 HouseCalc 时，因为使用 Call 语句所以需要利用括号将参数括起来。

```
Sub Main()
    HouseCalc 99800, 43100
    Call HouseCalc(380950, 49500)
End Sub

Sub HouseCalc(price As Single, wage As Single)
    If 2.5 * wage <= 0.8 * price Then
        MsgBox "You cannot afford this house."
    Else
        MsgBox "This house is affordable."
    End If
End Sub
```

在调用 Function 过程时，要想获得返回值，必须使用括号，必须

指定函数给变量，并且用括号将参数封闭起来，就可以使用函数的返回值。如下示例所示：

Answer3 = MsgBox("Are you happy with your salary?", 4, "Question 3")

如果不在意函数的返回值，可以用调用 Sub 过程的方式来调用函数。如下面示例所示，可以省略括号，列出参数并且不要将函数指定给变量：

Call MsgBox("Task Completed!", 0, "Task Box")

MsgBox "Task Completed!", 0, "Task Box"

当用这种方法调用函数时，Visual Basic 放弃返回值。

✍ 在上述最后的例子中若包含括号，则语句会导致一个语法错误。

其他模块的公用过程(Public)可以在工程的任何地方调用。如要调用其他模块中的过程，比如窗体中的过程、类模块中的过程、标准模块中的过程等等，就必须满足一些条件。

在窗体中的过程，那么所有窗体模块的外部调用都必须指向包含此过程的窗体模块。例如在窗体模块 Form1 中包含某个过程 AnySub，则可使用下面的语句调用 Form1 中的过程：

Call　Form1.AnySub(arguments)

在类模块中也要调用与过程一致并且指向该类模块的变量。但是，

与调用窗体过程不同，类模块在引用一个实例时，不能用类名称作为限定符。必须首先声明它为对象变量，再用变量名引用它。

例如，如果 FontClass 是 Class1 中的实例，要调用它：

```
Dim FontClass As New Class1
FontClass.AnySub
```

调用标准模块中的过程，如果过程名是唯一的，就不必在调用时加模块名。无论在模块内调用，还是在模块外调用，其结果都引用这个唯一的过程。如果两个或者两个以上的模块都包括相同名称的过程，就要用模块名来限定了。例如，模块 1 和模块 2 中名为 PriceName 的过程，从模块 2 中调用它，就会运行模块 2 中的 PriceName 过程，而不是模块 1 中的 PriceName 过程。其语句为：

```
Module2. PriceName(argument)
```

5.4.3 向过程传递参数

参数的数据类型当缺省时为 Variant（变体型）类型，也可以声明参数为其他的数据类型。在下例中说明了某个函数如何接收一个字符串。

例 12：

```
Function menu(Day As String)
'根据是否节日，返回不同的菜单。
```

```
    If Day="holiday" Then
        Menu="meet"
        '如果是节日，返回 meet。
    Else
        Menu="vegetable"
        '否则，返回蔬菜。
    EndIf
End Function
```

参数的传递方式一般有两种：

（1）按值传递参数：

按值传递时，实际上传递的是变量的副本。如果过程改变了这个值，则只影响副本而不影响变量本身。

用关键字 ByVal 指出变量按值传递。例如：

```
Sub PostPrice(ByVal PostNum As Integer)
…
…'这里写语句。
End Sub
```

（2）按地址传递参数：

按地址传递参数使过程用变量的内存地址去访问变量的实际内容。其结果是当变量传递给过程时，通过过程可以改变变量的值。

在 Visual Basic 中，按地址传递参数是缺省的。

如果在按地址传递参数时已指定其数据类型，那么就必须将这种类型的值传递给参数。可以给参数传递一个表达式，而不是数据类型。

把变量转换成表达式的最简单的方法就是把它放在括号内。

例 13：把已声明为整数的变量传递给过程，该过程以字符串为参数。

```
Sub CallExample()
    Dim IntX As Integer
    IntX=3*14
    Change (intX)
End Sub


Sub Change(Bar As String)
    MsgBox Bar
    'Bar 的值为字符串‘42’。
End Sub
```

下面介绍使用这些有可选参数的过程。在过程的参数表中列入关键字 Optional，就可以指定过程的参数是可选的。如果指定了可选参数，则参数表中的其他参数也必须是可选的，并且也要用 Optional 关键字来声明。

例 14：下面两段代码是假设有一个窗体，其内部有一个命令按钮和一个列表框。第一段代码提供所有可选参数。

```
Dim    Name    As    String
Dim    Adress    As    Variant

Sub    TextList(Optional x As String,Optional y As Variant)
    Text1.AssItem    x
    Text1.AssItem    y
End sub

Private    Sub    Command_Click()
    Name="yourname"
    Optional
    '提供了两个参数。
        Address=68154567
        Call TextList(Name,Address)
End Sub
```

这第二段代码并未提供全部可选参数。

```
Dim    Name    As    String
Dim    Adress    As    Variant
```

```
Sub TextList(x As String,Optional y As Variant)
    Text1.AssItem x
If Not IsMissing(y) Then
'用 IsMissing 函数来测试是否丢失可选参数。
    Text1.AssItem y
End If
End Sub

Private Sub Command_Click()
    Name="yourname"
    '并未提供第二个参数，即只提供了一个参数。
        Call TextList(Name,Address)
End Sub
```

也可以给可选参数指定缺省值。下面通过一个例子来介绍怎样提供可选参数的缺省值。

例 15：如果未将可选参数传递到函数过程，则返回一个缺省值。

```
Sub TextList(x As String,Optional y As Variant=68154567)
    Text1.AssItem x
    Text1.AssItem y
End sub
```

```
Private Sub Command_Click()
    Name="yourname"
    '未提供第二个参数。
    Call TextList(Name)
    '添加“yourname”和“68154567”。
End sub
```

一般来说，过程调用中的参数个数应该等于过程说明中的参数。但是，如果用关键字 ParamArray 指明，过程则可以接收任意个数的参数。

下面的例子是计算总和的 Sum 函数。

例 16：

```
Dim x As Variant
Dim y As Integer
Dim intSum As Integer

Sub Sum(ParamArray intNum())
  For Each x In intNums
      Y=y+x
  Next x
```

```
    IntSum=y
End Sub
Private Sub Command_Click()
    Sum 1,3,5,7,8
    List1.AddItem intSum
End Sub
```

最后，介绍一下用命名的参数创建简单语句。对于许多内部函数和语句，Visual Basic 提供了命名函数的方法来以快捷方式传递参数值。通过给命名的参数赋值，就可以按任意次序提供任意数目的参数。

Sub 或 Function 过程中的语句可以利用命名参数来传递值给被调用的过程。命名参数的组成是由参数名称紧接着冒号（:=）以及等号，然后指定一个值给参数。

下面的示例使用命名参数来调用不具返回值的 MsgBox 函数。

```
MsgBox Title:="Task Box", Prompt:="Task Completed!"
```

下面的示例使用命名参数调用 MsgBox 函数。将返回值指定给变量 answer3。

```
answer3 = MsgBox(Title:="Question 3", _
Prompt:="Are you happy with your salary?", Buttons:=4)
```

例 17：在下面的例子中，参数顺序与所要的参数顺序相反。

Function TextList(Name As String,Optional Address As

```
Variant)
        List1.AddItem Name
        List2.AddItem Address
    End   Sub

    Private   Sub   Command_Click()
        List Address:=68154567,Name:="yourname"
    End Sub
```

- 使用命名参数时，不能省略所需要的参数输入，可以只省略可选参数。对 Visual Basic 和 Visual Basic For Application 对象库，“对象浏览器”的对话框将可选参数用[ ]括起来。

为确定哪一个函数、语句、方法支持命名参数，用“代码”窗口中的“AutoQuickInfo”功能，检查“对象浏览器”，或者直接参阅“联机帮助”。

5.5　VBA 的内部函数

VBA 中可以使用以下几种内部函数：数学函数，字符串函数，日期和时间函数，类型转换函数和逻辑测试函数。下面列举各自的主要函数类别及功能。

5.5.1 数学函数

主要包括三角函数、指数对数函数、方根函数、随机函数等，如表 5-4。

表 5-4　数学函数

函数名称	函数功能
Sin	返回一个双精度数值，指定参数的正弦值。 语法是：Sin(number)，number 是 Double 或任何有效的数值表达式，表示一个以弧度为单位的角。 说明：Sin 函数取一角度为参数值，返回角的对边长度除以斜边长度的比值。结果的取值范围在-1 到 1 之间。为了将角度转换为弧度，将角度乘以 pi/180；为了将弧度转换为角度，将弧度乘以 180/pi。
Cos	返回一个双精度数值，指定参数的余弦值。 语法是：Cos(number)，number 是双精度数值或任何有效的数值表达式，表示一个以弧度为单位的角。Cos 函数返回角的邻边长度除以斜边长度的比值。其他方面与 Sin 函数基本相同。
Tan	返回一个双精度数值，指定参数的正切值。 语法是：Tan(number)，number 是双精度数值或任何有效的数值表达式，表示一个以弧度为单位的角。关于角的说明与 Sin 函数的基本相同。
Exp	返回双精度数值，指定 e（自然对数的底）的某次方。 语法：Exp(number)，其中 number 是双精度数值或任何有效的数值表达式。 说明：如果 number 的值超过 705.782712893，则会导致错误发生。常数 e 的值大约是 2.718282。 注意 ：Exp 函数的作用和 Log 的作用互补，所以有时也称做反对数。

续 表

函数名称	函数功能
Log	返回双精度数值，指定参数的自然对数值。 语法：Log(number)，其中 number 是双精度数值或任何有效的大于 0 的数值表达式。 说明：自然对数是以 e 为底的对数。常数 e 的值大约是 2.718282。 如果将 x 的自然对数值除以 n 的自然对数值，就可以对任意底 n 来计算数值 x 的对数值：Logn(x)=Log(x)/Log(n) 可以编写函数求以 10 为底的对数值： Static Function Log10(X) Log10(X) = Log(X)/Log(10) End Function
Sqr	返回一个双精度数值，指定参数的平方根。 语法：Sqr(number) 其中 number 参数是双精度数值或任何有效的大于或等于 0 的数值表达式。
Rnd	Rnd 函数返回小于 1 但大于或等于 0 的 Single 类型的随机数值。 语法：Rnd[(number)]，其中 number 的值决定了 Rnd 生成随机数的方式。 如果 number<0,那么每次使用 number 作为随机数种子得到的相同结果；如果 number>0，或者 number 值缺省，则生成序列中的下一个随机数；如果 number=0，则会生成最近生成的数。 为了生成某个范围内的随机整数，可使用以下公式： Int((upperbound - lowerbound + 1) * Rnd + lowerbound) 这里，upperbound 是随机数范围的上限，而 lowerbound 则是随机数范围的下限。 注意若想得到重复的随机数序列，在使用具有数值参数的 Randomize 之前直接调用具有负参数值的 Rnd。使用具有同

	样number值的Randomize一般是不会得到重复的随机数序列的。
Abs	返回参数的绝对值，其类型和参数相同。 语法：Abs(number)，其中的number是任何有效的数值或表达式。如果number包含Null，则返回Null，如果number是未初始化的变量，则返回 0。 一个数的绝对值是将正负号去掉以后的值。例如，ABS(-1) 和 ABS(1) 都返回 1。
Round	返回一个数值，该数值是按照指定的小数位数进行四舍五入运算的结果。 语法：Round(expression[,numdecimalplaces]) Round函数语法有两部分：其中expression是要进行四舍五入运算的数值表达式；而numdecimalplaces部分是可选的数字值，表示进行四舍五入运算时，小数点右边应保留的位数。如果忽略，则Round函数返回整数。
Fix Int	返回参数的整数部分。 语法：Fix(number)，Int(number)。 其中number是双精度数值或任何有效的数值表达式。如果number包含Null，则返回Null。 Int 和 Fix都会删除number的小数部分而返回剩下的整数。 Int 和 Fix的不同之处在于，如果number为负数，则Int返回小于或等于number的第一个负整数，而Fix则会返回大于或等于number的第一个负整数。例如，Int将-8.4转换成-9，而Fix将-8.4转换成-8。 Fix(number)等于：Sgn(number) * Int(Abs(number))
Sgn	返回一个Variant(Integer)，指出参数的正负号。 语法：Sgn(number)，其中的number参数是任何有效的数值表达式。 number参数的符号决定了Sgn函数的返回值。如果number<0，返回-1；如果number=0，返回0；如果number>0返回＋1。

除了这些纯数学函数之外，还有一些关于计算资产折旧、现金流及

支付利率等方面的应用函数，如 DDB、IPmt、PPmt、Pmt、IRR、MIRR、NPER、NPV、PV、FV、SYD、SLN、Rate 等。现在简要介绍它们的功能：

DDB：返回一个双精度数值，指定一笔资产在一特定期间内的折旧。可使用双下落收复平衡方法或其他指定的方法进行计算。

Pmt：返回一个双精度数值，指定根据定期定额支付且利率固定的年金支付额。

IPmt：返回一个双精度数值，指定在一段时间内对定期定额支付且利率固定的年金所支付的利息值。

PPmt：返回一个双精度数值，指定在定期定额支付且利率固定的年金的指定期间内的本金偿付额。

IRR：返回一个双精度数值，指定一系列周期性现金流（支出或收入）的内部利率。

MIRR：返回一个双精度数值，指定一系列修改过的周期性现金流（支出或收入）的内部利率。

PV：返回一个双精度数值 指定在未来定期、定额支付且利率固定的年金现值。

NPV：返回一个双精度数值，指定根据一系列定期的现金流（支付和收入）和贴现率而定的投资净现值。

NPER：返回一个双精度数值，指定定期定额支付且利率固定的总

期数。

FV：返回一个双精度数值，指定未来的定期定额支付且利率固定的年金。

SYD：返回一个双精度数值，指定某项资产在一个指定期间用年数总计法计算的折旧。

SLN：返回一个双精度数值，在一期里指定一项资产的直线折旧。

Rate：返回一个双精度数值，指定每一期的年金利率。

当然，其中的语法规则与注意事项还需要进一步学习，详细内容还可以参阅联机帮助中的“Visual Basic 语言参考”中的函数部分。

5.5.2 字符串函数

字符串函数主要针对字符串进行操作，包括比较，连接，搜索等。字符串函数的名称与作用的详细内容如表 5-5 所示。

表 5-5　字符串函数

函数名称	函数功能
String	返回 Variant(String)，指定长度的重复字符串。 语法：String(number,character)。 参数 number 是返回的字符串长度，如果它包含 Null，将返回 Null。而 Character 为指定字符的字符码或字符串表达式,其第一个字符将用于建立返回的字符串。另外，如果指定 character 的数值大于 255,String 会按下面的公式将其转为有效的字符码：character Mod 256。
Str	返回代表某一数值的字符串。

	语法：Str(number)，number 参数为 Long 类型，其中可包含任何有效的数值表达式。 说明：当某一数字转成字符串时，总会在前头保留一个空位来表示正负。如果 number 为正，返回的字符串包含一个前导空格暗示它是正号。 使用 Format 函数可将数值转成必要的格式，如日期、时间、货币或其他用户自定义格式。但是 Format 函数与 Str 不同的是 Format 函数不包含前导空格来放置 number 的正负号。
StrReverse	返回一个字符串，其中一个指定子字符串的字符顺序是反向的。 语法：StrReverse(expression) 参数 expression 是一个字符串，调用此函数后，它的字符顺序要被反向。如果 expression 是一个长度为零的字符串("")，则返回一个长度为零的字符串。如果 expression 为 Null，则产生一个错误。

续　表

函数名称	函数功能
StrConv	返回按指定类型转换的字符串。 语法：StrConv(string,conversion,LCID)各个参数的意义是： string 是要转换的字符串表达式；conversion，Integer 类型，其值的和决定转换的类型。LCID 可选的，如果与系统 LocaleID 不同，则为 LocaleID（系统 LocaleID 为缺省值。） StrConv 的几个常数值是： Visual BasicUpperCase 默认为 1，作用是将字符串文字转成大写。 Visual BasicLowerCase 默认为 2，可以将字符串文字转成小写。

	Visual BasicProperCase 值是 3，将字符串中每个字的开头字母转成大写。 Visual BasicWide*默认值是 4，将字符串中单字节字符转成双字节字符。 Visual BasicNarrow*值为 8，将字符串中双字节字符转成单字节字符。 Visual BasicKatakana**值为 16，将字符串中平假名字符转成片假名字符。 Visual BasicHiragana**值为 32，将字符串中片假名字符转成平假名字符。 Visual BasicUnicode 的值是 64，它根据系统的缺省码页将字符串转成 Unicode。注意在 Macintosh 中不可用。 Visual BasicFromUnicode 的值为 128，它将字符串由 Unicode 转成系统的缺省码页。注意在 Macintosh 中不可用。 这些常数是由 VBA 指定的。可以在程序中使用它们来替换真正的值。其中大部分是可以组合的，例如 Visual BasicUpperCase+Visual BasicWide，但是互斥的常数不能组合，例如 Visual BasicUnicode+Visual BasicFromUnicode。当在不适用的区域使用常数 Visual BasicWide、Visual BasicNarrow、Visual BasicKatakana，和 Visual BasicHiragana 时，就会导致运行时错误。
Asc	返回一个 Integer 类型，代表字符串中首字母的字符代码。 语法：Asc(string) ，其中的 string 参数可以是任何有效的字符串表达式。如果 string 中没有包含任何字符，则会产生运行时错误。 说明：在非 DBCS 系统下，返回值范围为 0 – 255 。在 DBCS 系统下，则为 -32768 – 32767。 另外 AscB 函数作用于包含在字符串中的字节数据，它返回第一个字节的字符代码，而非字符的字符代码。AscW 函数返回 Unicode 字符代码，若平台不支持 Unicode，则与 Asc 函数功能相同。
Left	返回 Variant (String)，其中包含字符串中从左边算起

	指定数量的字符。 语法：Left(string, length) 它有下面两个必要的命名参数： string：给定的字符串表达式。 Length：数值表达式。指出将返回多少个字符。 例如：Left("niceday",3)，将返回三个字符的字符串："nic"。
Right	与Left函数相反，它返回从字符串右边取出的指定数量的字符，类型也是Variant (String)。 语法：Right(string, length) Right 函数的命名参数与Left函数基本相同，只是从右端截取，例如：Right("niceday",3)将返回"day"。
Mid	返回 Variant (String)，其中包含字符串中从start开始的指定数量的字符。 语法：Mid(string, start[, length])它的参数有： string：字符串表达式，从中返回字符。 Start：为字符串中被取出部分的字符开始位置。如果start 超过总字符数，Mid 返回零长度字符串 ("")。 Length：可选参数，为 Variant (Long)，是返回的字符数。如果省略或length已超过文本的字符数(包括start处的字符)，将返回字符串中从start到尾端的所有字符。

续　表

函数名称	**函数功能**
Ltrim Rtrim Trim	这三个函数都返回 Variant (String)，是指定字符串的拷贝，但分别没有前导空白 (LTrim)、尾随空白 (RTrim)、前导和尾随空白 (Trim)。 语法：LTrim(string) RTrim(string) Trim(string) string 参数可以是任何有效的字符串表达式。如果string 包含 Null，将返回 Null。
LCase	返回转成小写的字符串。语法：LCase(string)，其中的

	string 参数可以是任何有效的字符串表达式。如果 string 包含 Null，将返回 Null。 说明：只有大写的字母会转成小写；所有小写字母和非字母字符保持不变。
UCase	返回转成大写的字符串。 语法：UCase(string)，其中参数为任何有效的字符串表达式。如果其中包含 Null，将返回 Null。 说明：只有小写的字母会转成大写；原本大写或非字母之字符保持不变。它与 LCase 是作用可逆的函数。
Space	返回特定数目空格的 Variant (String)。 语法：Space(number)，参数为字符串中想要的空格数。Space 函数在格式输出或清除固定长度字符串数据时很有用。
Len	返回字符串内字符的数目，或是存储一变量所需的字节数。 语法：Len(string ｜ varname)，其参数有： string：任何有效的字符串表达式。 varname：任何有效的变量名称。如果 varname 是 Variant 类型，Len 会视其为 String，总是返回其包含的字符数。 当在用户自定义数据类型中使用变长字符串时，Len 可能不能确定实际存储所需的字节数目。 注意在使用 LenB 函数时，返回的是用于代表字符串的字节数，而不是返回字符串中字符的数量。
Instr	返回指定一字符串在另一字符串中最先出现的位置。 语法 ：InStr([start,]string1, string2[, compare]) InStr 函数的参数有： Start：为可选参数，数值表达式，设置每次搜索的起点。如果省略，将从第一个字符的位置开始。如果包含 Null，将发生错误。如果指定了 compare 参数，则一定要有 start 参数。 String1：接受搜索的字符串表达式。 String2：被搜索的字符串表达式。 Compare：可选参数。指定两个字符串比较。如果 compare

	是 Null，将发生错误。如果省略 compare，Option Compare 的设置将决定比较的类型，指定一个有效的 LCID (LocaleID) 以在比较中使用与区域有关的规则。这里的 compare 与 StrComp 函数中的 compare 相同。 关于 InStr 的返回值： 当 string1 为零长度，返回 0。 string1 为 Null 时，返回 Null。 string2 为零长度时，返回 Start 的值。 string2 为 Null，返回 Null。 string2 找不到时，返回值为 0。 在 string1 中找到了 string2 时，返回找到的位置。 start>string2 时，返回值为 0。 另外，提到 InStrB 函数，它作用于包含在字符串中的字节数据。所以 InStrB 返回的是字节位置，而不是字符位置。

续　表

函数名称	函数功能
Oct	返回 Variant (String)，代表一个数值的八进制值。 语法：Oct(number)，其必要的 number 参数为任何有效的数值表达式或字符串表达式。 如果 number 尚非整数，那么在执行前会先四舍五入成最接近的整数； 如果 number 是 Null，返回值为 Null；如果 number 是 Empty,返回值为 0； 如果 number 为任何其他的值，最多可得 11 个字符。 也可以将适当范围的数前缀以 &O 来直接表示八进制数字。例如，八进制表示法的 &O10 代表十进制的 8。
Hex	返回代表十六进制数值的 String。 语法：Hex(number)，参数为任何有效的数值表达式或字符串表达式。

	如果 number 还不是一个整数，那么在执行前会先被四舍五入成最接近的整数。适当范围内的数字，前缀以 &H，可以直接表示十六进制数字。例如，十六进制表示法的 &H10 代表十进制的 16。 如果 number 是 Null，返回值为 Null；如果 number 是 Empty,返回值为 0；如果 number 为任何其他的值，最多可以得 8 个字符。
Join	返回一个字符串,该字符串是通过连接某个数组中的多个子字符串而创建的。 语法： Join(sourcearray[, delimiter]) Join 函数语法有如下命名参数： soursearray 是必需的，包含被连接子字符串的一维数组。 Delimiter 是可选的，在返回字符串中用于分隔子字符串的字符。如果忽略该项，则使用空格(" ")来分隔子字符串。如果 delimiter 是零长度字符串("")，则列表中的所有项目都连接在一起，中间没有分隔符。
Replace	返回一个字符串,该字符串中指定的子字符串已被替换成另一子字符串，并且替换发生的次数也是指定的。 语法： Replace(expression, find, replace[, start[, count[, compare]]]) Replace 函数语法有如下命名参数： Expression 是必需的字符串表达式，包含要替换的子字符串。 Find 是必需的要搜索到的子字符串。 Replace 是必需的用来替换的子字符串。 Start 是可选的在表达式中子字符串搜索的开始位置。如果忽略，假定从 1 开始 Count 是可选的子字符串进行替换的次数。如果忽略，缺省值是 -1，它表明进行所有可能的替换。 Compare 是可选的数字值，表示判别子字符串时所用的比较方式。关于其值，请参阅 StrComp 的“设置值”部分。

	Replace 的返回值如下设定： 当 expression 长度为零时，返回零长度字符串("")。 当 expression 为 Null 时，返回一个错误。 当 find 长度为零时，返回 expression 的复本。 当 replace 长度为零时，返回 expression 的复本，它删除了所有出现的 find 的字符串。 当 start>Len(expression)时，返回长度为零的字符串。 当 count=0 时，返回 expression 的复本。 Replace 函数的返回值是一个字符串，但是，其中的从 start 所指定的位置开始，到 expression 字符串的结尾处的一段子字符串已经发生过替换动作。并不是原字符串从头到尾的一个复制。

续　表

函数名称	函数功能
Format	返回 Variant (String)，其中含有一个表达式，它是根据格式表达式中的指令来格式化的。 语法：Format(expression[, format[, firstdayofweek[, firstweekofyear]]]) Format 函数的语法具有下面四个部分： Expression 是必要的任何有效的表达式。 Format 是可选参数，可以是有效的命名表达式或用户自定义格式表达式。 Firstdayofweek 是可选参数，常数，表示一星期的第一

	天。 Firstdayofyear 也是可选参数，常数，表示一年的第一周。 其中的 Firstdayofweek 的值有下面几种设置： 常数 Visual BasicUseSystem 值为 0，表示使用 NLS API 设置。 常数 Visual BasicSunday 值为 1，表示星期日（缺省）。 常数 Visual BasicMonday 值为 2，表示星期一。 常数 Visual BasicTuesday 值为 3，表示星期二。 常数 Visual BasicWednesday 值为 4，表示星期三。 常数 Visual BasicThursday 值为 5，表示星期四。 常数 Visual BasicFriday 值为 6，表示星期五。 常数 Visual BasicSaturday 值为 7，表示星期六。 另外 Firstofyear 的值有下面几种设置： Visual BasicFirstJan1 值为 1，表示从包含一月一日的那一周开始（缺省）。 Visual BasicFirstFourDay 值为 2，表示从本年第一周开始，而此周至少有四天在本年中。 Visual BasicFirstFullWeek 值为 3，表示从本年第一周开始，而此周完全在本年中。 关于格式化，对于数字，使用预先定义的命名数值格式或创建用户自定义数值格式；对于日期和时间，使用预先定义的命名日期/时间格式或创建用户自定义日期/时间格式；对于日期和时间序数，使用日期和时间格式或数值格式；对于字符串，创建自定义的字符串格式。 Format 与 Str 的区别在 Str 部分已有叙述。
Chr	返回 String，其中包含有与指定的字符代码相关的字符 。 语法：Chr(charcode)，其中的 charcode 参数是一个用来识别某字符的 Long 类型。 说明：0 到 31 之间的数字与标准的非打印 ASCII 代码相同。例如，Chr(10)可以返回换行字符。charcode 的正常范围为 0 – 255。在 DBCS 系统，charcode 的实际范围为 -32768 到 65535。

	注意：ChrB 函数作用于包含在 String 中的字节数据。ChrB 总是返回一个单字节，而不是返回一个字符，一个字符可能是一个或两个字节。 Visual Basic for the Macintosh 不支持 Unicode 字符串。因此，当 n 值在 128－65,535 的范围内时，ChrW(n) 不能像在 Windows 环境中那样返回所有的 Unicode 字符。相反，当 Unicode 的 n 值大于 127 时，ChrW(n)会试图做一个“最好的猜测”。因此，在 Macintosh 环境中，不能使用 ChrW 。
Val	返回包含于字符串内的数字，字符串中是一个适当类型的数值。 语法：Val(string) ，其中的 string 参数可以是任何有效的字符串表达式。 说明：Val 函数，在它不能识别为数字的第一个字符上，就停止读入字符串。那些被认为是数值的一部分的符号和字符，例如美圆号，不能被识别。但是函数可以识别进位制符号 &O（八进制）和 &H（十六进制）。空白、制表符和换行符都从参数中被去掉。 例如： Val(″　　1615 198th Street N.E.″)的返回值为 1615198。 Val 为所示的十六进制数值返回十进制数值。例如： Val(″&HFFFF″)返回值-1。 注意 Val 函数只会将句点（.）当成一个可用的小数点分隔符。当使用不同的小数点分隔符时，如在各个国家的应用程序中，代之以 CDbl 来把字符串转换为数字。

下面举例说明字符串函数的应用。

例 1：使用 String 函数来生成某个指定长度且只含单一字符的字符串。

```
Dim　StringType
StringType = String(8,"*")
```

'将返回"********"。

StringType = String(5,48)

'将返回"HHHHH"。

StringType = String(9,"xyz")

'将返回"xxxxxxxxx"。

例 2：Str 函数的例子，它将一个数字转成字符串。当数字转成字符串时，字符串的第一个位一定是空格或是正负号。

Dim MyString

MyString = Str(786)

'返回" 786"。

MyString = Str(-58.43)

'返回"-58.43"。

MyString = Str(14565.008)

'返回" 14565.008"。

例 3：StrComp 函数比较两个字符串。如果第三个参数值为 1，字符串是以文本比较的方式进行比较;如果第三个参数值为 0 或是缺省，则以二进制比较的方式进行比较。

✍ 文本比较方式会将大小写字母视为一样，但二进制比较方式则视为不同。

Dim Str1, Str2, Comp

Str1 = "ABC": Str2 = "abc"

'定义 Str1 和 Str2 及 Comp 变量。

Comp = StrComp(Str1, Str2, 1)

'以文本方式比较，认为 string1 等于 string2，所以返回 0。

Comp = StrComp(Str1, Str2, 0)

'以二进制方式比较，string1 小于 string2，所以返回-1。

MyComp = StrComp(Str2, Str1)

'以缺省方式比较，即以二进制方式比较，由于 string2 大于 string1，所以返回 1。

例 4：使用 Asc 函数返回字符串首字母的字符值（ASCII 值）。

Dim SomeNumber

SomeNumber = Asc("A")

'返回字符“A”的 ASCII 值 65。

SomeNumber = Asc("a")

'返回字符“a”的 ASCII 值 97。

SomeNumber = Asc("Apple")

'返回字符串的首字符“A”的 ASCII 值 65。

例 5：分别使用 Left 函数和 Right 函数来得到某字符串最左边和最右边的几个字符。

Dim SomeString,StrLeft,StrRight

SomeString="Stand up"

'定义字符串。

StrLeft=Left(SomeString,1)

'返回从左边数第一个字符"S"。

StrRight=Right(SomeString,1)

'返回从右边数第一个字符"p"。

StrLeft=Left(SomeString,7)

'返回从左边起的 7 个字符"Stand u"。

StrRight=Right(Something,7)

'返回从右边数的 7 个字符"tand up"。

StrLeft=Left(SomeString,20)

'返回"Stand up"。

StrRight=Right(SomeString,20)

'返回" Stand up"。

例 6：使用 Mid 函数来得到某个字符串中的几个字符。

Dim SomeString, Word1, Word2, Word3

SomeString="Long Live China"

'建立一个字符串。

Word1= Mid(SomeString,1,3)

'返回从第一个字符开始的 3 个字符，即"Lon"。

Word2= Mid(SomeString,10,5)

'返回从第 10 个字符开始的 5 个字符，即"China"。

Word3=Mid(SomeString,5)

'返回从第 5 个字符开始直到结束的所有字符，即"Live China"。

例 7：使用 LTrim 及 RTrim 函数将某字符串的开头及结尾的空格全部去除。事实上只使用 Trim 函数也可以做到将两头空格全部去除。

Dim SomeStr,TrimStr

SomeStr=" <-Help me-> "

'设置字符串初值。

TrimStr=LTrim(SomeStr)

'删除左边的空格，返回 TrimStr= "<-Help me-> "。

TrimStr= RTrim(SomeStr)

'删除右边的空格，返回 TrimStr=" <-Help me->"。

TrimStr= LTrim(RTrim(SomeStr))

'先删除右边空格，再删除左边空格，最终返回 TrimStr= "<-Help me->"。

'只使用 Trim 函数也同样将两头空格去除。

TrimStr=Trim(SomeStr)

'返回 TrimStr="<-Help me->"。

例 8：使用 LCase 函数来将某字符串转成全部小写，使用 UCase

函数来将某字符串转成全部大写。

Dim　Case,UpperCase,LowerCase

Case="Hello World aB12"

'要输送的字符串。

Lowercase=Lcase(Case)

'将所有大写变成小写，其他不变，返回"hello world ab12"。

UpperCase=UCase(Case)

'将所有小写变成大写，其他不变，返回"HELLO WORLD AB12"。

例 9：使用 InStr 函数来查找某字符串在另一个字符串中首次出现的位置。

Dim　SearchStr,SearchChar,Position

SearchStr="Mum mum　Daddy　daddy"

'定义被搜索的字符串。

SearchChar="d "

'要查找字符串"d"。

Positon=Instr(7,SearchStr,SearchChar,1)

'从第 7 个字符开始，以文本比较的方式找起。小写 d 和大写 D 在文本比较下是一样的,返回值为 10（大写 D）。

Position=Instr(1,SearchStr,SearchChar,0)

'从第一个字符开始，以二进制比较的方式找起。小写 d 和大写 D

在二进制比较下是不一样的，所以返回值为 12（小写 d）。

Position=Instr(SearchStr,SearchChar)

'缺省的比较方式为二进制比较，返回 12。

Position=Instr(1,SearchStr,"W")

'由于字符串内不存在 W，所以返回 0。

例 10：本示例使用 Oct 函数将某数值转换为 8 进制表达式，使用 Hex 函数来得到某数值的 16 进制值。

Dim　OctVal

OctVal=Oct(4)

'将 4 转换成 8 进制，返回 4。

OctVal=Oct(8)

'将 8 转换成 8 进制，返回 10。

OctVal=Oct(459)

'将 459 转换成 8 进制，返回 713。

Dim　HexVal

HexVal=Hex(5)

'将 5 转换成 16 进制，返回 5。

HexVal=Hex(10)

'将 10 转换成 16 进制，返回 A。

HexVal=Hex(459)

'将 459 转换成 16 进制，返回 1CB。

例 11：使用 Chr 函数来返回指定字符码所代表的字符。

```
Dim　CharVal
CharVal=Chr(97)
'返回 a。
CharVal=Chr(62)
'返回>。
CharVal=Chr(37)
'返回%。
```

例 12：使用 Val 函数返回字符串中所含的数值。

```
Dim　CharVal
CharVal=Val("2457")
'返回 2457。
CharVal=Val("2　4 57")
'返回 2457。
CharVal=Val("24 and 57")
'返回 24。
```

5.5.3　日期和时间函数

日期与时间的函数主要介绍关于 Visual Basic 的日期和时间的设置

与调整。详细内容如表 5-6 所示。

表 5-6　日期与时间函数

函数名称	函数功能说明
Date	返回包含系统日期的 Variant (Date)。 语法：Date 使用 Date 语句是为了设置系统日期。
Time	返回一个指明当前系统时间的 Variant (Date)。 语法：Time 使用 Time 语句可以设置系统时间。
Timer	返回一个 Single，代表从午夜开始到现在经过的秒数。 语法：Timer Microsoft Windows 中,Timer 函数返回一秒的小数部分。
TimeValue	返回一个时间的 Variant(Date)变量。 语法:TimeValue(time),time 通常是一个字符串表达式,表示 0:00:00(12:00:00 A.M.)到 23:59:59(11:59:59 P.M.)之间的时刻。time 也可以是表示在同一时间范围取值的任何其他表达式。如果 time 包含 Null，则返回 Null。 可以使用 12 小时制或 24 小时制的时间格式。但是，如果 time 包含日期信息，TimeValue 将不会返回它。若 time 包含无效的日期信息，则会导致错误发生。
MonthName	返回一个表示指定月份的字符串。 语法：MonthName(month[, abbreviate]) 其中 month 是月份的数值表示。例如一月是 1，二月是 2；而 abbreviate 是可选的 Boolean 值，表示月份名是否缩写。如果忽略，缺省值为 False，表明月份名不能被缩写。

续　表

函数名称	函数功能说明

DateSerial	返回包含指定的年、月、日的 Variant(Date)。 语法：DateSerial(year,month,day)。 参数均为整型，其中 Year 的范围是从 100 到 9999。也可以用数值表达式表示某日之前或其后的年、月、日数。 例如：DateSerial(1990-10,8-2,1-1) 这里,DateSerial 函数返回 1990 年 8 月 1 日的十年零两个月又一天之前的日期，就是 1980 年 5 月 31 日。 year 参数的数值若介于 0 与 29 之间，则将其解释为 2000–2029 年，若介于 30 和 99 之间则解释为 1930–1999 年。而对所有其他 year 参数，则请用四位数值表示（如 1800）。 当任何一个参数的取值超出可接受的范围时，它会适时进位到下一个较大的时间单位。例如，如果指定了 35 天，则这个天数被解释成一个月加上多出来的日数，多出来的日数将由其年份与月份来决定。但是如果一个参数值超出-32768 到 32767 的范围，就会导致错误发生。
DateValue	返回一个 Variant(Date)。 语法：DateValue(date)，其中的参数 date 通常是字符串表达式，表示从 100 年 1 月 1 日到 9999 年 12 月 31 日之间的一个日期。 如果 date 是一个字符串，且其内容只有数字以及分隔数字的日期分隔符，则 DateValue 就会根据系统中指定的短日期格式来识别月、日、年的顺序。例如：12/30/1991 和 12/30/91。DateValue 也可识别明确的英文月份名称，全名或缩写均可，例如它可以识别 December 30, 1991 和 Dec 30, 1991。 如果 date 中略去了年这一部分，DateValue 就会使用由计算机系统日期设置的当前年份。如果 date 参数包含时间信息，则 DateValue 不会返回它。但如果 date 包含无效时间信息（如 89:98），则会导致错误发生。
DateAdd	返回包含一个日期的 Variant(Date),这一日期还加上了一段时间间隔。 语法：DateAdd(interval, number, date) ，其中 interval 是必要的字符串表达式，是所要加上去的时间

	间隔。Number 是必要的数值表达式，是要加上的时间间隔的数目，可以为正数（得到未来的日期），也可以为负数(得到过去的日期)。Date 是必要的 Variant (Date) 或表示日期的文字，这一日期还加上了时间间隔。 Interval 的参数值的设置为：yyyy 表示年，q 表示季，m 表示月，y 表示一年的日数，d 表示日，w 表示一周的日数，ww 表示周，h 表示小时，n 表示分钟，s 表示秒。 可以使用 DateAdd 函数对日期加上或减去指定的时间间隔。例如，可以用 DateAdd 来计算距今天为三十天的日期；或者计算距现在为 45 分钟的时间。但是有时，DateAdd 函数将不返回有效日期。举例说明： DateAdd(m,1,31-Jan-99) 此例是将 1999 年 1 月 31 日加上一个月，DateAdd 将返回 1999 年 2 月 28 日，而不是 1999 年 2 月 31 日。如果参数 date 是 2000 年 1 月 31 日，由于 2000 年是闰年，返回值是 2000 年 2 月 29 日。
DateDiff	返回 Variant(Long)的值,表示两个指定日期间的时间间隔数目。 语法： DateDiff(interval,date1,date2[,firstdayofweek[,firstweekofyear]])。DateDiff 函数语法中的 interval 参数与 DateAdd 中的意义和设定值相同；date1 和 date2 是计算中要用到的两个日期；firstofweek 与 firstofyear 可选，分别表示一个星期的第一天的常数（如果未予指定，则以星期日为第一天）和一年的第一周的常数 DateDiff 函数可用来决定两个日期之间所指定的时间间隔数目。
Day	返回一个其值为 1 到 31 之间的整数，表示一个月中的某一日。 语法：Day(date)，其中的 date，可以是任何能够表示日期的 Variant、数值表达式、字符串表达式或它们的组合。

续　表

函数名称	函数功能说明
DatePart	返回一个包含已知日期的指定时间部分的 Variant (Integer)。 语法：DatePart(interval, date[,firstdayofweek[, firstweekofyear]]) DatePart 函数可以用来计算日期并返回指定的时间间隔。例如，可以使用 DatePart 计算某个日期是星期几或目前为几点钟。firstdayofweek 参数会影响使用时间间隔符号“W”或“WW”计算的结果。如果 date 是日期文字，则指定的年份成为该日期的固定部分。但是，如果 date 用双引号("")括起来，且年份略而不提，则在每次计算 date 表达式时，当前年份都会插入到代码之中。这样就可以书写适用于不同年份的程序代码。
TimeSerial	返回一个具有具体时、分、秒的时间。 语法：TimeSerial(hour,minute,second) 其中，为了指定一个时刻，TimeSerial 的参数取值应在正常范围内：钟点应介于 0-23 之间，分钟与秒应介于 0-59 之间。当然，也可以为使用数值表达式的参数指定相对时间。例如：TimeSerial(12-6,-15,0) 就使用了表达式代替绝对时间数。函数返回中午之前六小时　又十五分钟的时间，即 5:45:00 A.M.。 当任何一个参数的取值超出正常范围时，它会适时进位到下一个较大的时间单位。例如，如果指定了 75（75 分钟)，则这个时间被解释成一小时又十五分。如果一个参数值超出 -32，768 到 32，767 的范围，就会导致错误发生。如果三个参数指定的时间会使日期超出可接受的日期范围，则亦会导致错误发生。

Month	返回一个Variant(Integer),其值为1到12之间的整数，表示一年中的某月。 语法：Month(date)， date可以是任何能够表示日期的Variant、数值表达式、字符串表达式或它们的组合。
Year	返回 Variant (Integer)，包含表示年份的整数。 语法：Year(date) date参数是任何能够表示日期的 Variant、数值表达式、字符串表达式或它们的组合。
Second	返回一个Variant(Integer),其值为0到59之间的整数，表示一分钟之中的某个秒。 语法：Second(time)，time参数，可以是任何能够表示时刻的Variant、数值表达式、字符串表达式或它们的组合。
Minute	返回一个Variant(Integer),其值为0到59之间的整数，表示一小时中的某分钟。 语法：Minute(time)，time参数是任何能够表示时刻的Variant、数值表达式、字符串表达式或它们的组合。
Hour	返回一个 Variant (Integer)，其值为 0 到 23 之间的整数，表示一天之中的某一钟点。语法：Hour(time) time可以是任何能够表示时刻的 Variant、数值表达式、字符串表达式或它们的组合。
Weekday	返回一个代表某个日期是星期几的整数。 语法Weekday(date, [firstdayofweek]) Weekday函数中的firstdayofweek参数是可选的，其设定值参照Format函数。date参数能够表示日期的Variant、数值表达式、字符串表达式或它们的组合。 Weekday函数的返回值也可参照Format函数。
Now	返回一个 Variant (Date)，根据计算机系统设置的日期和时间来指定日期和时间。 语法：Now
WeekdayName	返回一个字符串，表示一星期中的某天。 语法：WeekdayName(weekday, abbreviate, firstdayofweek)

	其中 weekday 是必需的数字值，表示一星期中的某天。该数字值要依赖于 firstdayofweek 设置中的设置值来决定；abbreviate 是可选的 Boolean 值，表示星期的名称是否被缩写。如果忽略该值，缺省值为 False，表明星期的名称不能被缩写；firdtdayofweek 是可选的数字值，表示一星期中第一天。关于其值，请参阅 Format 函数的“设置值”部分。

5.5.4　逻辑测试函数

逻辑测试函数的返回值是 True 或 False。具体内容如表 5-7 所示。

表 5-7　逻辑测试函数

函数名称	函数功能说明
IsArray	返回 Boolean 值，作用是指出变量是否为一个数组。 语法：IsArray(varname)，varname 参数是一个指定变量的标识符。 如果变量是数组，则 IsArray 返回 True；否则返回 False。对于包含数组的 variant 表达式来说，IsArray 尤为有用。
IsDate	返回 Boolean 值，指出一个表达式是否可以转换成日期。 语法：IsDate(expression)，expression 参数是一个 Variant，包含日期表达式或字符串表达式，这里的字符串表达式是可以作为日期或时间来认定的。 如果表达式是一个日期，或可以作为有效日期识别，则 IsDate 返回 True；否则返回 False。有效日期的范围介于公元 100 年 1 月 1 日与公元 9999 年 12 月 31 日之间；其有效范围随操作系统不同而不同。
IsEmpty	返回 Boolean 值，指出变量是否已经初始化。 语法：IsEmpty(expression)，expression 参数是一个 Variant，包含一个数值或字符串表达式。expression 参数通常是单一变量名。如果变量未初始化或已明确设置为 Empty，则 IsEmpty 返回 True；否则返回 False。如果

	expression 含有多个变量，则 IsEmpty 总是返回 False。IsEmpty 只返回对 variant 表达式有意义的信息。
IsError	返回 Boolean 值，指出表达式是否为一个错误值。语法：IsError(expression)，expression 参数是任何有效表达式。利用 CVErr 函数将实数转换成错误值就会建立错误值。IsError 函数被用来确定一个数值表达式是否表示一个错误。如果参数表示一个错误，则 IsError 返回 True；否则返回 False。
IsNull	返回 Boolean 值，指出表达式是否不包含任何有效数据(Null)。 语法：IsNull(expression)，参数 expression 是一个包含数值表达式或字符串表达式的 Variant。如果 expression 为 Null，则 IsNull 返回 True；否则 IsNull 返回 False。如果 expression 由多个变量组成，则表达式的任何作为变量组成成分的 Null 都会使整个表达式返回 True。 Null 值指出 Variant 不包含有效数据。Null 与 Empty 不同，后者指出变量尚未初始化。Null 与长度为零的字符串(“”)也不同，长度为零的字符串指的是空串。
IsNumber	返回 Boolean 值，指出表达式的运算结果是否为数。 语法：IsNumeric(expression)。其中的 expression 是一个包含数值表达式或字符串表达式的变体型变量。如果整个 expression 的运算结果为数字，则 IsNumeric 返回 True；否则返回 False。如果 expression 是日期表达式，则 IsNumeric 返回 False。
IsObject	返回 Boolean 值，指出标识符是否表示对象变量。 语法:IsObject(identifier)。identifier 是一个变量名。如果 identifier 是 Object 类型或任何有效的类的类型，或者 identifier 是 VarTypeVisual BasicObject 的 Variant 或用户自定义的对象，则 IsObject 返回 True；否则返回 False。即使变量已设置成 Nothing，IsObject 也仍返回 True。 使用错误捕获方法可以确认对象引用是否有效。
IsMissi	返回 Boolean 值，指出一个可选的 Variant 参数是否已经

ng	传递给过程。 语法：IsMissing(argname)。argname 参数包含一个可选的 Variant 过程参数名。 使用 IsMissing 函数可以检测在调用一个程序时是否提供了可选 Variant 参数。如果对特定参数没有传递值过去，则 IsMissing 返回 True;否则返回 False。如果 IsMissing 对某个参数返回 True，则在其他代码中使用这个丢失的参数将产生一个用户自定义的错误。如果对 ParamArray 参数使用 IsMissing，则函数总是返回 False。为了检测空的 ParamArray，可试看一下数组的上界是否小于它的下界。 注意：IsMissing 对简单数据类型（例如 Integer 或双精度数值）不起作用，因为与 Variant 不同，它们没有“丢失”标志位的前提。正由于此，对于可选参数类型，可以指定缺省值。如果调用过程时，参数被忽略，则该参数将具有该缺省值。

下面列举几个简单例子说明它们的应用。

例 1：使用 IsMissing 函数检查是否把可选参数传送给用户自定义过程。

注意：除了 Variant 类型以外，其他皆可当作 Optional 参数的缺省值及类型。

Dim　ReturnVal

ReturnVal=Twice()

'上面的语句调用了用户自定义函数。

ReturnVal=Twice(6)

'返回 12。

'下面是函数过程定义。

Function　ReturnTwice(Optional　X)

　　If　IsMissing(X)　Then

　　　　Twice=Null

'如果参数丢失，则返回 Null。

　　Else

　　　　Twice=X*2

'如果参数出现，则返回两倍的值。

　　End　If

End　Funciton

例 2：使用 IsArray 函数来检查变量是否为数组。

Dim　MyArr(1To10)　As　Integer,YourArr,MyCheck

'声明数组变量。

YourArr=Array(1,2,3)

'使用数组函数。

MyCheck=IsArray(MyArr)

'返回 True。

MyCheck=IsArray(YourArr)

'返回 True。

例 3：使用 IsEmpty 函数检查变量是否已经初始化。

Dim　MyVar,MyCheck

MyCheck = IsEmpty(MyVar)

'返回 True。

MyVar=Null

'赋以 Null。

MyCheck=IsEmpty(MyVar)

'返回 False。

MyVar = Empty

'赋以 Empty。

MyCheck = IsEmpty(MyVar)

'返回 True。

例 4：使用 IsNull 函数检查变量值是否为 Null。

Dim　MyVar,MyCheck

MyCheck=IsNull(MyVar)

'返回 False。

MyVar=""

MyCheck = IsNull(MyVar)

'返回 False。

MyVar=Null

MyCheck=IsNull(MyVar)

'返回 True。

5.5.5　类型转换函数

类型转换函数是在变量类型之间进行相互转换的函数。具体内容如表 5-8 所示。

表 5-8　类型转换函数

函数名称	函数功能说明
CBool	语法：CBool(expression)，expression 为任何有效的字符串或数值表达式。CBool 函数强制将表达式转换成 Boolean 类型。
CCur	语法：CCur(expression)，参数范围为 -922,337,203,685,477.5808 至 922,337,203,685,477.5807 的数值。CCur 函数把表达式强制转换成 Currency 类型。
CDbl	语法是：CDbl(expression)。 其参数的范围是： 负数从 -1.79769313486231E308 至 -4.94065645841247E-324； 正数从 4.94065645841247E-324 至 1.79769313486232E308。 它将表达式强制转换成双精度数值类型。
CInt	Cint 函数将表达式转换成整数型。表达式的值可以从 -32,768 至 32,767，小数部分四舍五入。 当小数部分恰好为 0.5 时,Cint 和 CLng 函数会将它转换为最接近的偶数值。例如，0.5 转换为 0、1.5 转换为 2。Cint 和 CLng 函数不同于 Fix 和 Int 函数，Fix 和 Int 函数会将小数部分截断而不是四舍五入。并且 Fix 和 Int 函数总是返回与传入的数据类型相同的值。
CLng	将表达式强制转换成长整型，表达式的值可以 -2,147,483,648 至 2,147,483,647，小数部分四舍五入。

CSng	将表达式强制转化成单精度浮点型(Single)。 表达式的取值范围是： 负数为-3.402823E38 至-1.401298E-45； 正数为 1.401298E-45 至 3.402823E38。
CStr	强制转化成字符串类型，依据 expression 参数返回 Cstr。
CVar	将表达式转化成变体型。若转化为数值，则参数范围与双精度数值 相同；若转化结果不为数值，则范围与 String 相同。
CByte	将表达式转化成字节类型，参数范围 0 至 255。
CDec	将表达式转化成 Decimal 类型。表达式范围： 零变比数值（无小数位） +/-79,228,162,514,264,337,593,543,950,335。 28 位小数的数值+/-7.9228162514264337593543950335。 最小的可能非零值是 0.0000000000000000000000000001。

续　表

函数名称	函数功能说明
CVErr	返回 Error 子类型的 Variant，其中包含指定的错误号。 语法：CVErr(errornumber)，errornumber 参数是任何有效的错误号代码。 可以在过程中，使用 CVErr 函数来创建用户自定义错误。例如，如果创建一个函数，它可以接受若干个参数，且正常返回一个字符串，则可以让函数来判断输入的参数，确认它们是在可接受的范围内。如果不是的话，此函数将不会返回所要的字符串。在这种情况下，CVErr 可以返回一个错误号，并告知应该采取的行动。 注意，Error 的隐式转换是不允许的，例如，不能直接把 CVErr 的返回值赋值给一个非 Variant 的变量。然而，

	可以对 CVErr 的返回值进行显式转换（使用 CInt、CDbl 等等），并赋值给适当的数据类型变量。
Cdate	语法：CDate(expression)，参数是任何有效的日期表达式。此函数强制将表达式转换成 Date 类型。如果使用 IsDate 函数，可判断 date 是否可以被转换为日期或时间。而 Cdate 可用来识别日期文字和时间文字，以及落入可接受的日期范围内的数值。当转换一个数字成为日期时，是将整数部分转换为日期，小数部分转换为从午夜起算的时间。 CDate 依据系统上的区域设置来决定日期的格式。如果提供的格式为不可识别的日期设置，则不能正确判断年、月、日的顺序。另外，长日期格式，若包含有星期的字符串，也不能被识别。 CVDate 函数也提供对早期 Visual Basic 版本的兼容性。CVDate 函数的语法与 CDate 函数是完全相同的，不过，CVDate 是返回一个 Variant，它的子类型是 Date，而不是实际的 Date 类型。因为现在已有真正的 Date 类型，所以 CVDate 也不再需要了。转换一个表达式成为 Date，再赋值给一个 Variant，也可以达到同样的效果。也可以使用这种技巧将其他真正的数据类型转换为对等的 Variant 子类型。

5.5.6 其他函数

此外，VBA 中还有一些重要函数，在此举例并加以详细介绍对话框函数 MsgBox 和 InputBox。附加简介一些其他函数。

（1）MsgBox 函数

其功能是在对话框中显示消息，等待用户单击按钮，并返回一个 Integer 告诉用户单击哪一个按钮。

语法：MsgBox(prompt[,buttons][,title][,helpfile,context])，其中的参

数详细意义和作用如表 5-9 所示。

表 5-9　MsgBox 函数的参数

函数名称	函数功能说明
Prompt	作为显示在对话框中的消息字符串表达式。最大长度大约为 1024 个字符，如果 prompt 的内容超过一行，则可以在每一行之间用回车符(Chr(13))、换行符(Chr(10))或是回车与换行符的组合(Chr(13) & Chr(10))将各行分隔开来。
Buttons	数值表达式是值的总和，指定显示按钮的数目及形式，使用的图标样式，缺省按钮是什么以及消息框的强制回应等。如果省略，则 buttons 的缺省值为 0。
Title	在对话框标题栏中显示的字符串表达式。如果省略 title，则将应用程序名放在标题栏中。
Helpfile	识别用来向对话框提供上下文相关帮助的帮助文件的字符串表达式。如果提供了 helpfile，则也必须提供 context。
Context	数值表达式，由帮助文件的作者指定给适当的帮助主题的帮助上下文编号。如果提供了 context，则也必须提供 helpfile。

buttons参数有下列设置值，如表 5-10 所示。

表 5-10　Button 参数的设置值

常数	值	描述
Visual BasicOKOnly	0	只显示 OK 按钮。
Visual BasicOKCancel	1	显示 OK 及 Cancel 按钮。
VBAbortRetryIgnore	2	显示 Abort、Retry 及 Ignore 按钮。
Visual BasicYesNoCance	3	显示 Yes、No 及 Cancel 按钮。

l		
Visual BasicYesNo	4	显示 Yes 及 No 按钮。
Visual BasicRetryCancel	5	显示 Retry 及 Cancel 按钮。
Visual BasicCritical	16	显示 Critical Message 图标。
Visual BasicQuestion	32	显示 Warning Query 图标。
Visual BasicExclamation	48	显示 Warning Message 图标。
Visual BasicInformation	64	显示 Information Message 图标。
Visual BasicDefaultButton1	0	第一个按钮是缺省值。
Visual BasicDefaultButton2	256	第二个按钮是缺省值。
Visual BasicDefaultButton3	512	第三个按钮是缺省值。
Visual BasicDefaultButton4	768	第四个按钮是缺省值。
VBApplicationModal	0	应用程序强制返回；应用程序一直被挂起，直到用户对消息框作出响应才继续工作。
Visual BasicSystemModa	4096	系统强制返回；全部应用程序都被挂起，直到用户对消息框作出响应才继续工

l		作。
Visual BasicMsgBoxHelpButton	16384	将 Help 按钮添加到消息框
Visual BasicMsgBoxSetForeground	65536	指定消息框窗口作为前景窗口
Visual BasicMsgBoxRight	524288	文本为右对齐
Visual BasicMsgBoxRtlReading	1048576	指定文本应为在希伯来和阿拉伯语系统中的从右到左显示

第一组值(0–5)描述了对话框中显示的按钮的类型与数目；第二组值(16,32,48,64)描述了图标的样式；第三组值(0,256,512)说明哪一个按钮是缺省值；而第四组值(0,4096)则决定消息框的强制返回性。将这些数字相加以生成 buttons 参数值的时候，只能由每组值取用一个数字。

✍ 这些常数都是 Visual Basic for Applications (VBA)指定的。可以在程序代码中到处使用这些常数名称，而不必使用实际数值。

Msgbox 函数的返回值如表 5-11 所示。

表 5-11　Msgbox 函数的返回值

常数	值	描述
Visual BasicOK	1	OK
Visual	2	Cancel

BasicCancel		
VBAbort	3	Abort
Visual BasicRetry	4	Retry

续　表

常数	值	描述
Visual BasicIgnore	5	Ignore
Visual BasicYes	6	Yes
Visual BasicNo	7	No

例如在以后的 VBA 应用中可以使用 MsgBox 函数，在具有“是”及“否”按钮的对话框中显示一条严重错误信息。示例中的缺省按钮为“否”，MsgBox 函数的返回值视用户按哪一个钮而定。

在还未介绍语句与过程之前，首先介绍几个简单的例子的演示结果。图 5-1 显示了一个调用 Msgbox 函数的例子。

例如：



图 5-1　msgbox 函数的例子

按钮(button)的值包括描述按钮的类型与数目、描述了图标的样式、说明默认按钮的缺省值、决定对话框的值。将这些数字相加以生成 buttons 参数值的时候，只能由每组值取用一个数字。剩下的四组是加入帮助和上下文的信息。

另外，在提供了 helpfile 与 context 的时候，用户可以按 F1(Windows) or HELP (Macintosh) 来查看与 context 相应的帮助主题。像 Microsoft Excel 这样一些主应用程序也会在对话框中自动添加一个 Help 按钮。

如果对话框显示 Cancel 按钮，则按下 ESC 键与单击 Cancel 按钮的效果相同。如果对话框中有 Help 按钮，则对话框中提供有上下文相关的帮助。但是，直到其他按钮中有一个被单击之前，都不会返回任何值。

> ✍ 如果要指定第一个命名参数以外的参数，则必须在表达式中使用 MsgBox。为了省略某些位置参数，必须加入相应的逗号分界符。

（2）Input 函数

返回 String，它包含以 Input 或 Binary 方式打开的文件中的字符。

语法：Input(number, [#]filenumber)

Input 函数的语法具有以下几个部分：number 是任何有效的数值表达式，指定要返回的字符个数；filenumber 是任何有效的文件号。

通常用 Print#或 Put 将 Input 函数读出的数据写入文件。Input 函数

只用于以 Input 或 Binary 方式打开的文件。与 Input#语句不同，Input 函数返回它所读出的所有字符，包括逗号、回车符、空白列、换行符、引号和前导空格等。

对于以 Binary 访问类型打开的文件，如果试图用 Input 函数读出整个文件，则会在 EOF 返回 True 时产生错误。在用 Input 读出二进制文件时，要用 LOF 和 Loc 函数代替 EOF 函数，而在使用 EOF 函数时要配合以 Get 函数。

这里仍举出几个简单示例，以便于理解和简单应用入门。图 5-2 显示一个调用 Input 函数的的例子。

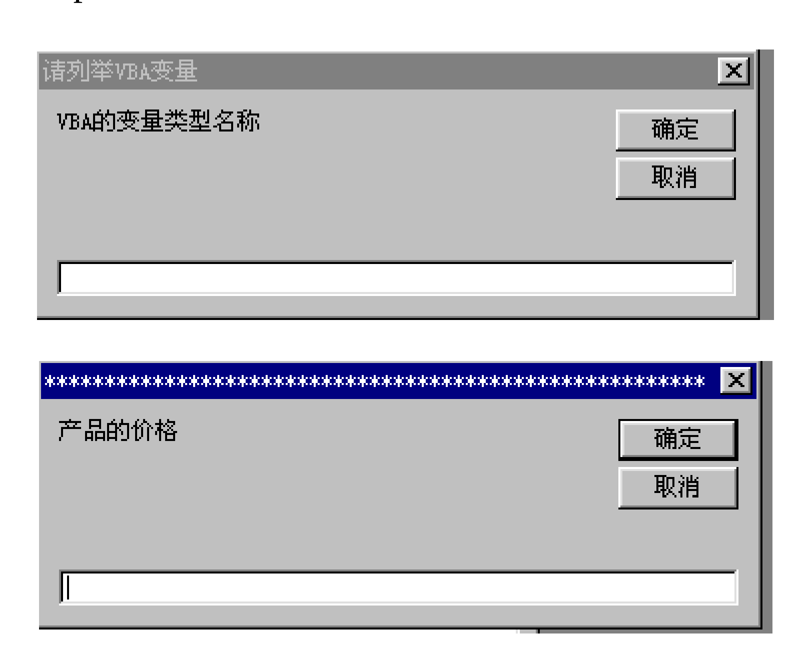


图 5-2　Input 函数的例子

（3）其他函数简介

IIf 函数的作用是根据表达式的值，来返回两部分中的其中一个。

CallByName 函数是执行一个对象的方法，或者设置或返回一个对象的属性。

Choose 函数可以从参数列表中选择并返回一个值。

Command 函数将返回命令行的参数部分。

CreateObject 函数创建并返回一个对 ActiveX 对象的引用。

CurDir 函数用来代表当前的路径。

DoEvents 函数转让控制权，以便让操作系统处理其他的事件。

Environ 函数关连于一个操作系统环境变量。

EOF 函数表明已经到达为 Random（随机）或顺序 Input 打开的文件的结尾。

Error 函数返回对应于已知错误号的错误信息。

FileAtter 函数表示使用 Open 语句所打开文件的文件方式。

FileDateTime 函数返回一个文件被创建或最后修改后的日期和时间。

FileLen 函数代表一个文件的长度，单位是字节。

Filter 函数返回一个下标从零开始的数组，该数组包含基于指定筛选条件的一个字符串数组的子集。

FreeFile 函数返回一个 Integer，代表下一个可供 Open 语句使用的文件号。

GetAllSettings 函数从 Windows 注册表或(Macintosh 中)应用程序初

始化文件中的信息中返回应用程序项目的所有注册表项设置及其相应值（开始是由 SaveSetting 产生）。

GetAttr 函数返回一个 Integer，此为一个文件、目录、或文件夹的属性。

GetObject 函数返回文件中的 ActiveX 对象的引用。

GetSetting 函数从 Windows 注册表中或 (Macintosh 中)应用程序初始化文件中的信息的应用程序项目返回注册表项设置值。

IMEStatus 函数返回一个 Integer，用来指定当前 Microsoft Windows 的输入法(IME)方式；只对东亚区版本有效。

LBound 与 UBound 函数均返回 Long 型数据。对于 Lbound，其值为指定数组维可用的下界；对于 Ubound，其值为指定的数组维可用的上界。

Loc 函数返回一个 Long，在已打开的文件中指定当前读/写位置。

LOF 函数返回一个 Long，表示用 Open 语句打开的文件的大小，该大小以字节为单位。

MacID 函数用在 Macintosh 上，将长为 4 个字符的常量转换成被 Dir，Kill，Shell 和 AppActivate 使用的值。

MacScript 函数执行一个脚本并返回由此脚本返回的值，如果脚本有返回值的话。

Partiton 函数返回一个 Variant(String)，指定一个范围，在一系列计

算的范围中指定的数字出现在这个范围内。

QBColor 函数返回一个 Long，用来表示所对应颜色值的 RGB 颜色码。

RGB 函数返回一个 Long 整数，用来表示一个 RGB 颜色值。

Shell 函数执行一个可执行文件，返回 Variant (Double)，如果成功的话，代表这个程序的任务 ID，若不成功，则会返回 0。

Spc 函数与 Print # 语句或 Print 方法一起使用，对输出进行定位。

Switch 函数计算一组表达式列表的值，然后返回与表达式列表中最先为 True 的表达式所相关的 Variant 数值或表达式。

Tab 函数与 Print #语句或 Print 方法一起使用，对输出进行定位。

TypeName 函数返回一个 String，提供有关变量的信息。

VarType 函数返回一个 Integer，指出变量的子类型。

第六章　VBA 的开发环境（IDE）与错误处理

在介绍了数据类型、函数和语句、过程等 Visual Basic 基础知识之后，本章将要展开利用 Visual Basic 来开发由简单到复杂的应用程序。在这一章，将深入介绍用户界面的样式、窗体、菜单、工具栏、内部控件及其用法，以及调试代码和处理错误等内容。

6.1　VBA 用户界面

在 Word2000 的 VBA 中，Visual Basic 编辑器是用来建立和管理 VBA 项目（VBA Project）的。在 Visual Basic 编辑器中，主要提供了工程资源管理器、代码窗口、属性窗口等调试环境，以帮助用户建立和管理应用程序。

Visual Basic 编辑器如图 6-1 所示。

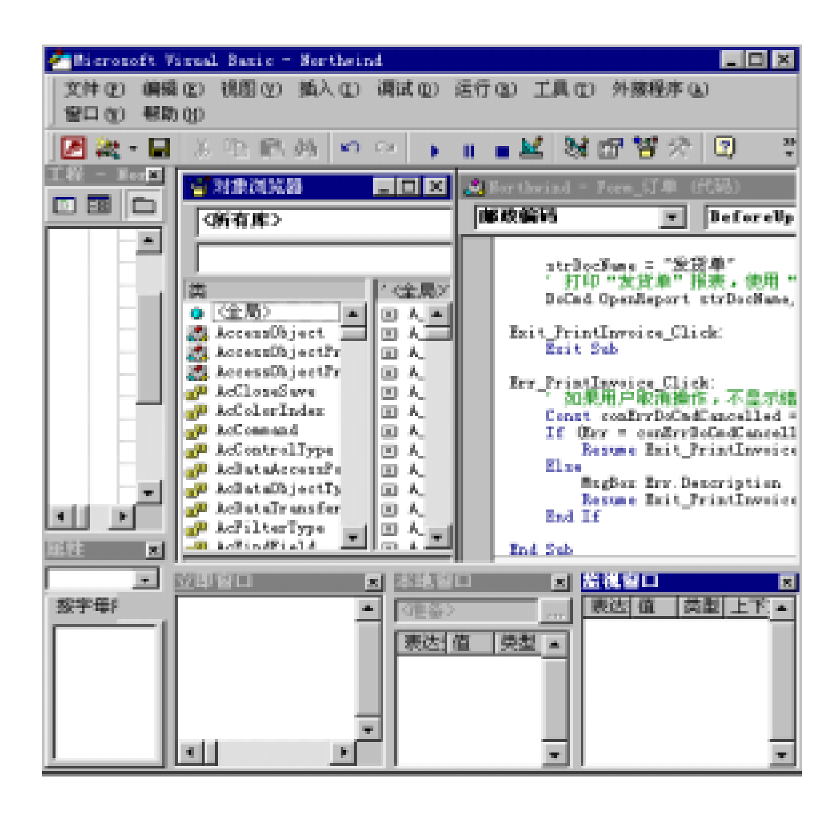


图 6-1　Visual Basic 编辑器

用户界面是一个应用程序最重要的部分。无论花多少精力编制代码，应用程序的可用性都依赖于程序的界面。

设计一个应用程序之前，需要做出有关界面的若干规定。比如使用单文档还是多文档样式，需要多少窗体，菜单中将包含什么命令，提供什么样的对话框与用户交互，需要提供多少帮助等。

在设计应用程序之前，还要考虑应用程序的目的。比如，用来显示信息的应用程序与用来收集信息的应用程序的目的显然不同；经常使用的应用程序与偶尔使用的应用程序也应该显然不同。

而用户的层次也影响设计。如果针对初学者用户，应用程序的设

计应该简单明确，针对有经验的用户，就可以设计得复杂一些。而且，用户使用的应用程序的方式也会影响它的设计，比如，程序需要发布到全球，那么语言和文化也是必须考虑到的部分。

建议用户界面的设计最好是作为一个反复进行的过程，因为很难在第一遍提出完美的设计。本章将在介绍 Visual Basic 界面设计中，同时介绍为用户创建应用程序的开发工具。

下面就分别详细介绍 Visual Basic 编辑器中的代码窗口、监视窗口、属性窗口、工程资源管理器、用户窗体等等。

6.1.1 代码窗口

可以使用代码窗口来编写、显示以及编辑 Visual Basic 代码。

打开“代码”窗口的效果如图 6-2 所示。

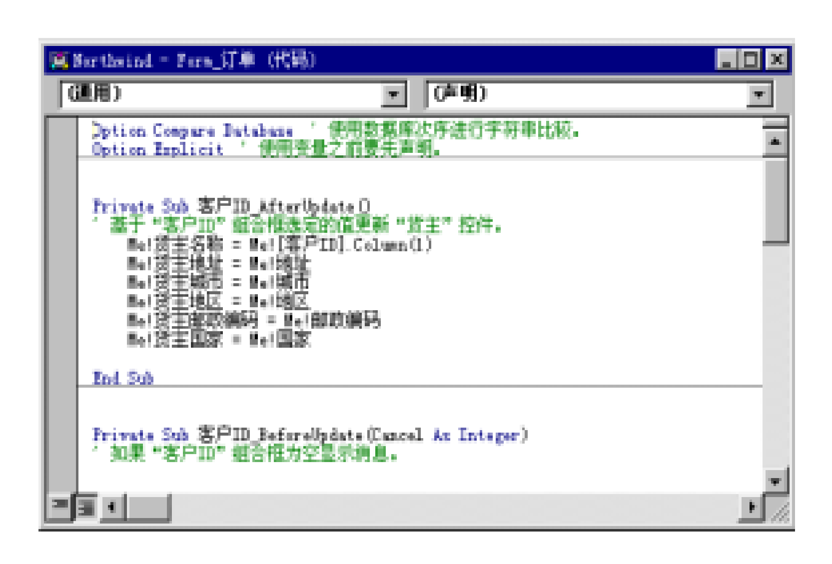


图 6-2　“代码”窗口

打开各模块的代码窗口后，可以查看不同窗体或模块(所谓模块，

就是一组声明集合，其后为过程)中的代码，并且可以在它们之间做复制以及粘贴的动作。

在代码窗口中，还可以建立或编辑宏程序、事件过程以及特殊的帮助程序。

可以按照下列所述的几种方式，来打开代码窗口，即用户进行以下几种操作时，代码窗口将会自动显示出来。

（1）在工程资源管理器中，可以选择一个代码对象双击。

（2）在 Visual Basic 编辑器窗口中，从“视图”菜单中选择“代码窗口”按钮，或者按下 F7 键。

（3）在“用户窗体”窗口中，可以双击控件或窗体。所谓控件，就是可放置在窗体上的对象，其中有它自己的属性、方法、事件。可用控件来接收用户的输入、显示输出、触发事件过程。可用方法操作大部分控件。有一些控件为交互作用式的(响应用户动作)，而有些则为静态的(仅能用代码访问)。

（4）在 Office 中编辑宏。

在打开代码窗口后，还可以将所选中的文本拖动到以下几个位置：

- 当前代码窗口中的不同位置。
- 其他的代码窗口。
- 立即窗口以及监视窗口中。
- “回收站”中。

关于窗口部件的构成，有以下几个部分。

（1）“对象”框：显示所选对象的名称。可以按列表框中的右边箭头（如图 6-2 的“声明”部分的箭头），来显示此窗体中的对象。

（2）“过程/事件”框：在窗体或对象框所含控件中，可以列出所有 Visual Basic 的事件。当选择了一个事件，则与事件名称相关的事件过程，就会显示在代码窗口中。“过程/事件”框实际上是按列表框中的左边箭头进行显示的（如图 6-2 中的“通用”一栏）。

如果在对象框中显示的是“通用”，则过程框会列出所有声明，以及为此窗体所创建的常规过程。如图 6-2 所示。

如果正在编辑模块中的代码，则过程框会列出所有模块中的常规过程。

在上述两例中，在过程框中所选的过程都会显示在代码窗口中。

模块中的所有过程会出现在单一滚动条的列表中，它们是按名称的字母来排列的。可以从代码窗口上端的下拉式列表中选取一个过程，此时指针会移到所选过程的第一行代码上面。例如，如图 6-3 所示，所有过程均位于过程框列表中，要显示某个过程的代码，直接从“过程/事件”列表框中选取它即可。

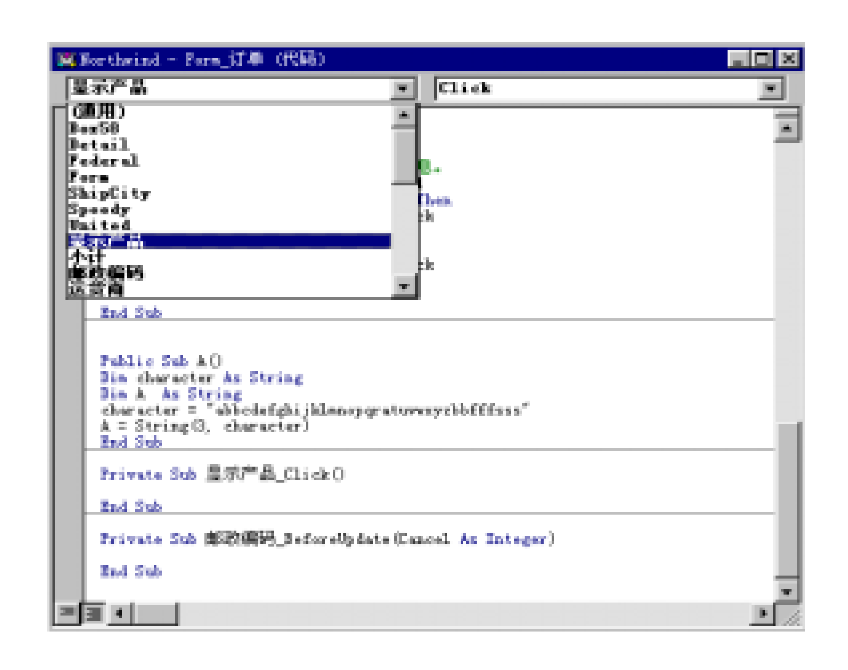


图 6-3　代码窗口中的过程框列表

（3）拆分栏：拆分栏是将“对象框”和“过程/事件框”的列表内容进行显示的按钮。如图 6-3 中的“显示产品”右端和“Click”右端的小三角形按钮。

只要将拆分栏向下拖放，将代码窗口分隔成两个都具有滚动条的水平窗格，就可以在同一时间查看当前代码中的不同部分，显示在“对象框”以及“过程/事件框”中的信息。

当前代码是指拥有焦点的窗格之内的代码。所谓焦点，是指具有在任何时间接收鼠标单击或键盘输入的能力。在 Microsoft Windows 环境中，在同一时间只有一个窗口、窗体或控件具有这种能力。“具有焦点”的对象通常会以突出显示标题或标题栏来表示。用户或应用程序可设置焦点。

将拆分栏拖放到窗口的顶部或下端，或者双击拆分栏，都可以关闭一个窗格。

另外，代码窗口的左边的灰色区域，在此会显示出边界标识。边界标识是代码窗口的边界标识条中显示的图标。在编辑代码期间，页边距指示区可提供一些视觉上的帮助。

在代码窗口的左下角，显示了两个图标：

“过程查看”图标：，显示所选的过程，同一时间只能在代码窗口中显示一个过程。

“全模块查看”图标：，显示模块中全部的代码。

图 6-3 中显示的“全模块查看”图标，即显示了模块中的全部代码。

图 6-4 则显示所选过程代码，即“过程查看”图标下的代码窗口。

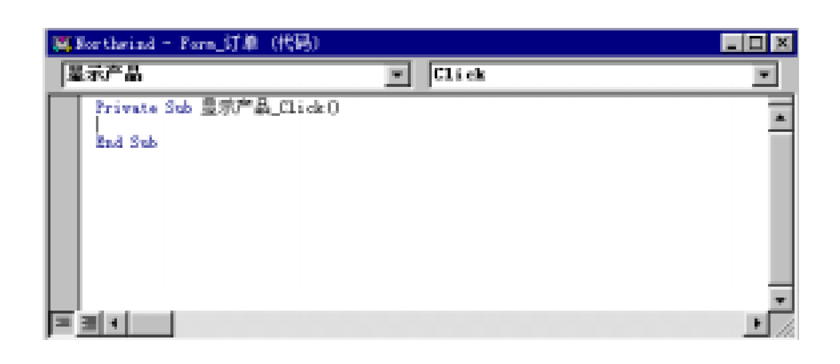


图 6-4　“过程查看”图标下的代码窗口

介绍完代码窗口的各个部件后，可以将主要精力投入到代码窗口的主体部分。

当在代码窗口中要输入的 VBA 代码涉及到对象或者数据类型时，

窗口会显示一个消息列表框，用来帮助用户输入正确的对象或数据类型。例如，当输入“Dim priceX As”时，代码窗口会出现一个显示所有对象和数据类型的列表框。如图 6-5 所示，用户可以选择 Double 等数据类型。

当代码窗口中，用户要输入的 VBA 代码涉及到函数名、子程序，代码窗口会显示一个提供函数的参数、数据类型及参数顺序的列表框，以提示用户输入正确的函数或子程序。例如，在使用函数 String 时，会显示如图 6-6 的列表，框内包括该函数的参数名称、参数类型、个数与顺序等信息。

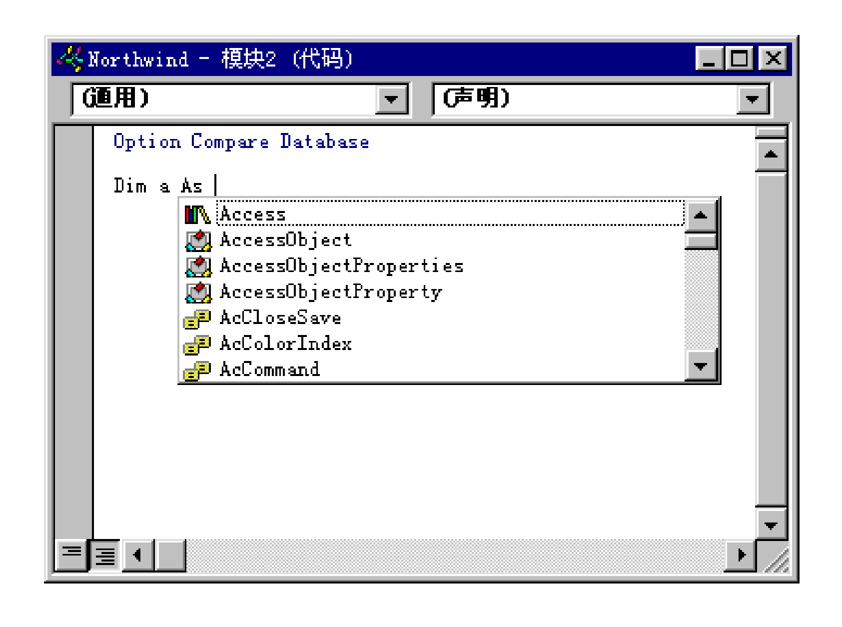


图 6-5　代码窗口的对象和数据类型列表

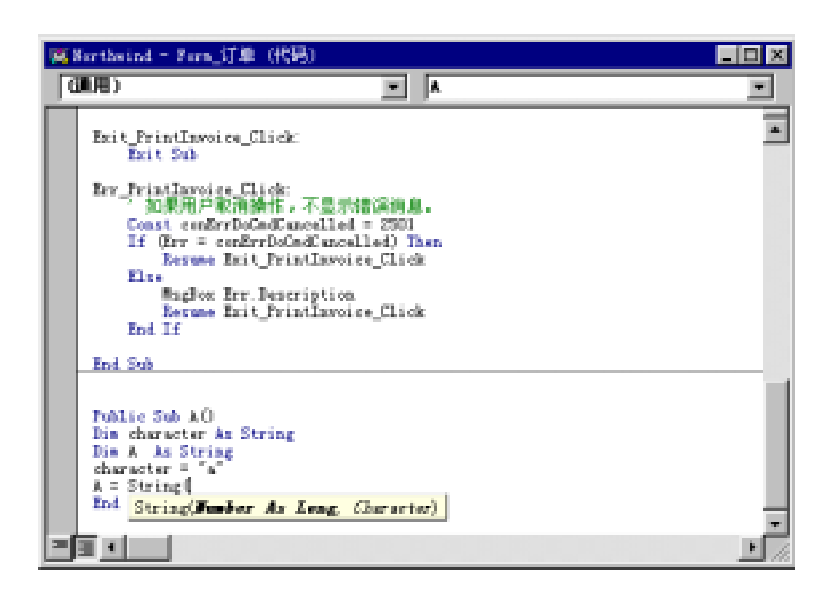


图 6-6　代码窗口的函数或子程序中的列表框

6.1.2　监视窗口

当工程中有定义监视表达式时，就会自动出现。也可以在视图菜单中选取监视窗口得到监视窗口。

监视窗口出现时的效果，如图 6-7 所示。

在监视窗口中，可以进行以下操作：

（1）重置列标头的大小，通过往右拖移边线来使它变大，或往左拖移边线来使它变小。

（2）拖动一个选取的变量，到立即窗口或监视窗口中。

（3）可以按下关闭框，来关闭一个窗口。如果关闭框不是可见的，可以先双击窗口标题行，让窗口变成可见。



图 6-7　监视窗口

从图 6-7 中的监视视图可以看到，监视窗口中有以下两部件组成：

（1）“表达式”：列出监视表达式，并在最左边列出监视图标。

（2）“值”：列出在切换成中断模式时表达式的值。

可以编辑一个值，并且按下 Enter 键，向上键，向下键，Tab 键，Shift+Tab 键或用鼠标在屏幕上单击，使编辑生效。如果这个值是无效的，则编辑字段会保持在作用中，并且值会以突出显示，且会出现一个消息框来描述这个错误。可以按下 Esc 键来中止更改。

（3）“类型”：列出表达式的类型。

（4）“上下文”：列出监视表达式的内容。

如果在进入中断模式时，监视表达式的内容不在范围内，则当前的值并不会显示出来。

6.1.3　属性窗口

在“编辑器”中选取“属性”窗口，回出现如图 6-8 所示的对话框。

属性窗口可以列出选取对象的属性的“设计时”以及当前设置的“设计时”的属性，还可以在“设计时”改变这些属性。当选取了多

个控件时，属性窗口会列出所有控件都具有的属性。

所谓“设计时”，就是开发环境中编译应用程序的时期，此时，添加控件、设置控件或窗体属性等。而在运行时，则像用户一样与应用程序交互作用。

属性窗口的主要部件包括：

（1）对象框

列出当前所选的对象，但只能列出现用窗体中的对象。如图 6-8 中的属性窗口的第一行的“订单　Form_订单”就是一个所选对象。

✍ 如果选取了好几个对象，则会以第一个对象为准，列出各对象均具有的属性。

（2）属性列表

分为“按字母序”选项卡和“按分类序”选项卡两种方式。

属性 - 订单

订单 Form_订单

按字母序 | 按分类序

(名称)	订单
AfterDelConfirm	
AfterInsert	
AfterUpdate	
AllowAdditions	True
AllowDeletions	True
AllowDesignChanges	False
AllowEdits	True
AllowFilters	True
AutoCenter	True
AutoResize	True
BeforeDelConfirm	
BeforeInsert	
BeforeUpdate	
BorderStyle	2
Caption	订单
CloseButton	True
ControlBox	True
Count	43

图 6-8 “按字母序”的属性窗口



图 6-9 “按分类序”的属性窗口

“按字母序”选项卡是按字母顺序列出所选的对象的所有属性，这些对象及其当前设置可以在设计时改变。

图 6-8 中的例子就是采用了“按字母序”的属性列表。可以看到，它的第一栏是所选对象“订单 Form_订单”中所包含的属性名称，第二栏是属性类型。

若要改变属性的设定，可以选择属性名，然后在第二栏键入新的类型；或直接在第二栏单击，从出现的三角形的拆分栏中选取新的设定。

“按分类序”选项卡是根据性质列出所选对象的所有属性。

可以折叠这个列表，这样将只看到分类；也可以扩充一个分类，并可以看到其所有的属性。当扩充或折叠列表时，可在分类名称的左边看到一个加号（+）或减号（-）图标。“按分类序”的属性列表如图 6-9 所示。

图 6-9 中显示在“格式”前面的是（-）图标，表示扩充了“格式”这一分类。事实上，这个例表的分类有“格式”、“事件”、“其他”、“数据”四类，当折叠这一列表，即各个分类前面是（+）时，即可看到这四类。按照类别列表，可以根据性质列出所选对象的所有属性，例如图 6-8 中的 AutoCEnter、Caption 和 BorderStyle 都属于外观格式上的属性，所以分类列表于图 6-9 的“格式”一栏，而 AfterInsert 和 AfterDelConfirm 属于“事件”一类，AllowAdditions 属于“数据”一

类，而 AllowDesignChanges 就属于“其他”一类等。

6.1.4　立即窗口

从“视图”菜单打开“立即”窗口，其效果如图 6-10 所示。

可以通过以下两种方式执行立即窗口的代码：

（1）键入或粘贴一行代码，然后按下 Enter 键来执行该代码。

（2）从立即窗口中复制并粘贴一行代码到代码窗口中，但是立即窗口中的代码是不能存储的。

立即窗口可以拖放到屏幕中的任何地方，除非已经在“选项”对话框中的“可连接的”选项卡内，将它设定为停放窗口。可以按下关闭框来关闭一个窗口。如果关闭框不是可见的，可以先双击窗口标题行，让窗口变成可见的。

✍ 注意：在中断模式下，立即窗口中的语句，是根据显示在过程框的内容或范围来执行的。举例来说，如果键入 Print variablename，则输出的就是局域变量的值。



图 6-10　立即窗口

6.1.5　本地窗口

在本地窗口中，可自动显示出所有在当前过程中的变量声明及变量值。图 6-11 是一个显示本地窗口的效果。



图 6-11　本地窗口

若本地窗口是可见的，则每当从执行方式切换到中断模式或是操纵堆栈中的变量时，它就会自动的重建显示。

所谓中断模式，就是在开发环境中暂时中止程序的执行。在中断模式下，可以检查、调试、重置、单步执行或继续执行程序。

在以下几种情况中可进入中断模式：

（1）在执行程序时遇到断点。

（2）在执行程序时按下 Ctrl+Break。

（3）在执行程序时遇到 Stop 语句或未捕获的运行时错误。

（4）添加一个 Break When True 监视表达式，当监视的值改变时将停止执行且值为 True。

（5）添加一个 Break When Changed 监视表达式，当监视的值改变时将停止执行。

下面介绍本地窗口的主要部件。

（1）“调用堆栈”按钮：它用来打开“调用堆栈”对话框，它会列出调用堆栈中的过程。

（2）“表达式”：在此栏列出变量的名称。

列表中的第一个变量是一个特殊的模块变量，可用来扩充显示出当前模块中的所有模块层次变量。对于类模块，会定义一个系统变量<Me>。对于常规模块，第一个变量是<name of the current module>。全局变量以及其他工程中的变量，都不能从本地窗口中访问。

✍ 不能在此栏编辑数据。

（3）“值”：在这一栏中列出所有变量的值。

当按下“值”字段中的一个值，指针就会变成“I”形，且值会被点划线包围。可以编辑一个值，并且按下 Enter 键，向上键，向下键，Tab 键，Shift+Tab 键或用鼠标在屏幕上单击，使编辑生效。如果这个值是无效的，则编辑字段会保持在作用中，且值会突出显示，并且会出现一个错误信息框来提示，此时可以按下 Esc 键来中止更改。

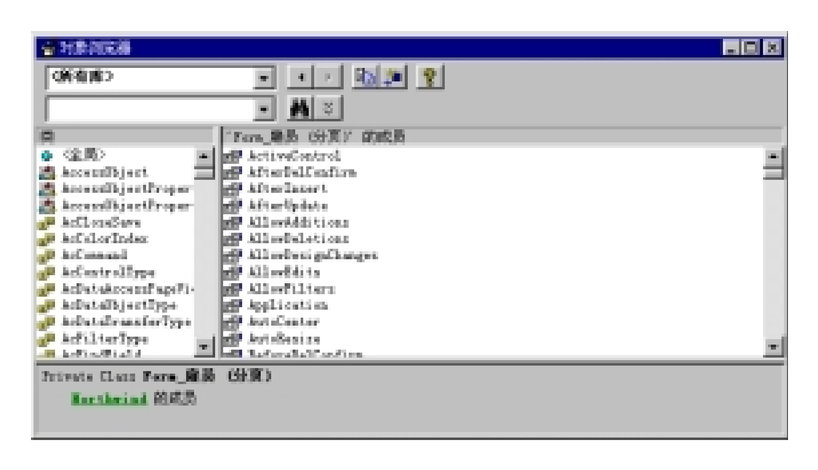
所有的数值变量都应该有一个值，而字符串变量则可以有空值。

拥有子变量的变量可以被扩充或折叠起来。折叠起来的变量不会显示一个值，而变量每一次会显示一个值。(+)以及(-)会出现在变量的左边。

（4）“类型”：在此栏列出变量的类型，也不能在此栏编辑数据。

6.1.6　对象浏览器

在视图菜单中选取“对象浏览器”，显示出的效果如图 6-12。



.图 6-12　对象浏览器

对象浏览器可以显示出对象库以及工程里的过程中的可用类、属性、方法、事件及常数变量。

可以用它来搜索及使用既有的对象，或是来源于其他应用程序的对象。

从图 6-12 中可以看出，其窗口部件包括以下几个部分：

（1）“工程/库”框

图 6-12 中标有<所有库>的列表框即为“工程/库”框。它会显示活动工程的当前所引用的库。可以在“引用”对话框中添加库。<所有库>可以一次显示出所有的库。

（2）“搜索文本”框

图 6-12 中的“工程/库”框下面一行，即为“搜索文本”框。它包含要用来做搜索的字符串。可以键入或选择所要的字符串。搜索文本框中包含最后四次输入的搜索字符串，直到关闭此工程为止。在键入字符串时，可以使用标准的 Visual Basic 通配符。

如果要查找完全相符的字符串，可以用快捷菜单中的“全字匹配”命令。

（3）“向后”按钮

可以向后回到前一个类及成员列表。每单击一次后便向前进至下一个选项，直到文档的最后。

（4）“向前”按钮

每次单击可以重复原本选择的类及成员列表，直到选择列表用完。

（5）“复制到剪贴板”按钮

将成员列表中的选择或详细框中的文本复制到剪贴板，然后在此后将选择贴到代码中。

（6）“视图定义”按钮

将光标移到“代码”窗口中，定义成员列表或类列表中选定的位

置。

（7）“帮助”按钮

显示在类或成员列表中，选定工程的联机帮助主题。也可以使用F1键。

（8）“搜索”按钮

激活类或属性、方法、事件或常数等符合您在“搜索文本”框中键入字符串的库搜索，并且打开有适当信息列表果会缺省的按类型创建组并从A到Z排列。

（9）“显示/隐藏搜索结果”按钮

打开或隐藏“搜索结果”框。“搜索结果”框改变成显示从“工程/库”列表中所选出的工程或库的搜索结果。

（10）“搜索结果列表”

如在“搜索文本”框中输入变量名A_ADD，再单击“搜索”按钮，就在对象浏览器中插入一个“搜索结果列表”。

它显示搜索字符串所包含工程的对应库、类及成员。“搜索结果”框在您改变“工程/库”框中的选择时改变。如图6-13所示。

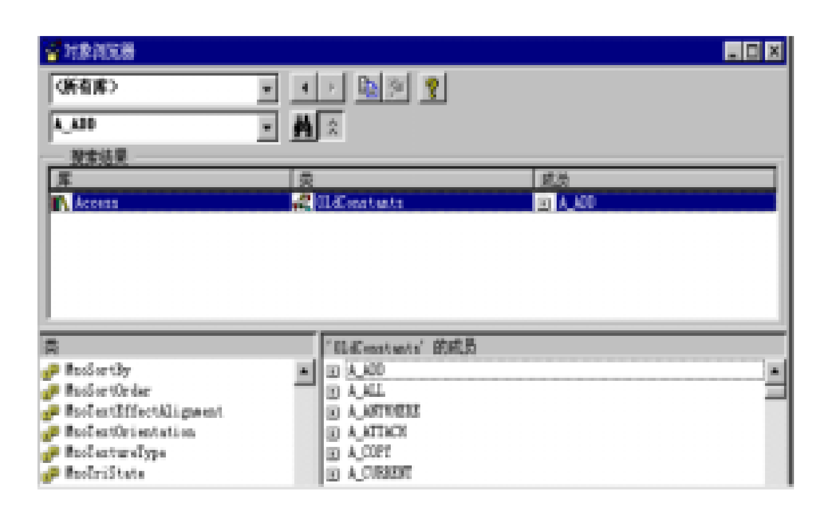


图 6-13　对象浏览器中的搜索结果列表显示

（11）“类列表”

它位于图 6-12 中的“搜索文本”框的下方，主要显示在“工程/库”框中选定的库或工程中所有可用的类。如果有代码编写的类，则这个类会以粗体方式显示。这个列表的开头都是<globals>，是可全局访问的成员列表。

对象浏览器及代码窗口包含许多代表类及成员的图标。

如果选择了类但没有选择特定的成员，就会得到缺省成员。缺省的成员以星号(*)或是以此成员特定的缺省图标（如表 6-1 中所列出的那些图标及所代表意义）做为标识。例如，图 6-13 中的“MsoSortBy”类是 Enum（枚举类型），成员“A_ADD”是常数，“Access”是一个“库”；而图 6-12 中的“AccessObject”代表一个类，“<全局>”则代表 Global，即全局变量的意义。

表 6-1 是图标的列表及代表什么意义。

表 6-1　对象浏览器中的图标及意义

图标	代表的意义
	属性
	缺省属性
	缺省方法
	事件
	常数
	模块
	类
	用户自定义类型
	Global
	库
	工程
	内置关键字及形态
	Enum

（12）“成员列表”

如图 6-12 和图 6-13 中的“类列表”的右端的列表项。它按组显示出在“类”框中所选类的元素，在每个组中再按字母排列。用代码编写的方法、属性、事件或常数会以粗体显示。可用“对象浏览器”的快捷菜单中的“组成员”命令改变此列表顺序。

（13）“详细数据”

显示成员定义。“详细数据”框包含一个跳转，以跳到该元素所属的类或库。某些成员的跳转可跳到其上层类。

例如，如图 6-14 所示，在“类列表”和“成员列表”的下方，显示的就是“详细数据”。单击“OldConstants 成员列表”中的任一常数，例如 A_DELETE_V2，就可以显示关于它的详细信息：

“Const A_DELETE_V2=7，Access OldConstants 的成员”。

还可以将“详细数据”框中的文本复制[;Win;>或拖动<]到“代码”窗口。

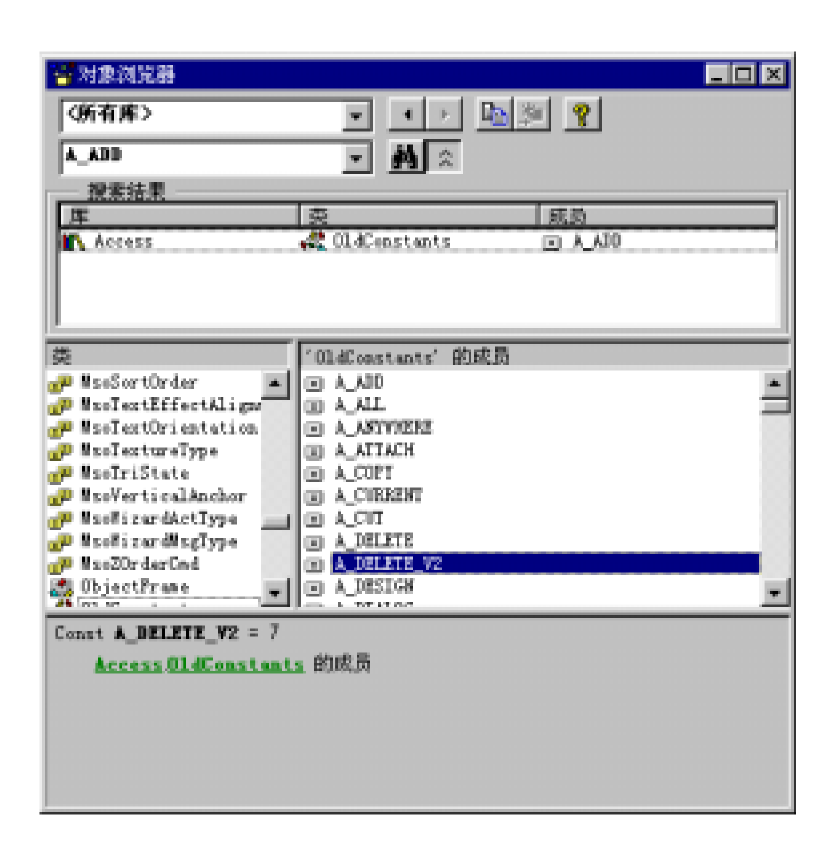


图 6-14　对象浏览器的成员列表及详细数据

（14）“拆分条”

拆分可以调整相应的框的大小。它位于“类”框及“成员”框之间、“搜索结果”列表及“类”与“成员”框之间，“类”与“成员”框及“详细数据”框之间。

6.1.7　工程资源管理器

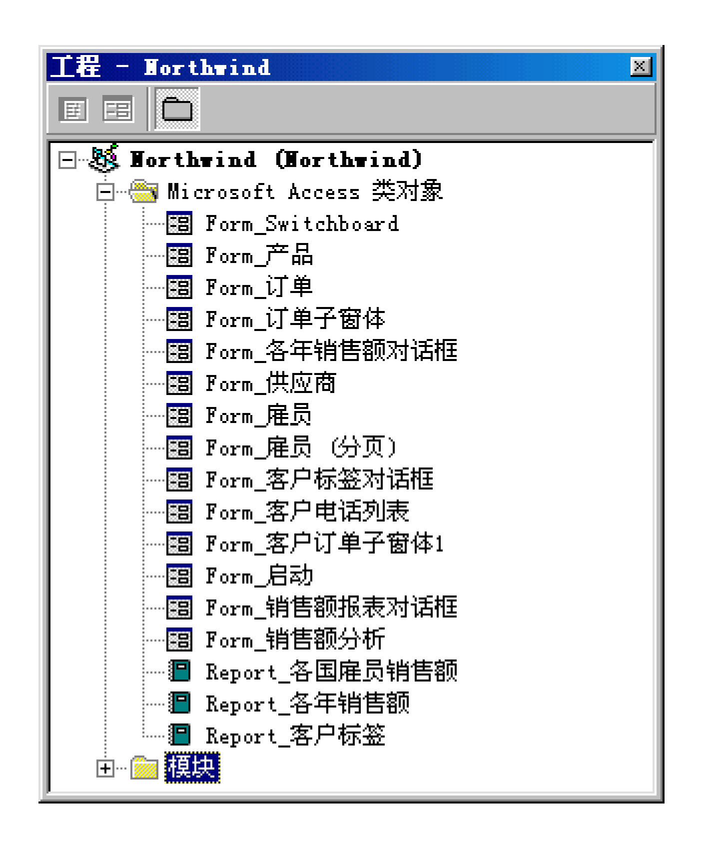


图 6-15　工程资源管理器

工程资源管理器显示工程的一个分层结构列表以及所有包含在此工程内的或者被引用的全部工程。

其显示结果如图 6-15 所示。

其窗口主要的部件有：

（1）“查看代码”

如图 6-15 中的图标，它位于“工程”栏的下方第一个，作用是显示代码窗口，以编写或编辑所选工程目标代码。

例如，选中图 6-15 中的“Form_订单”，再点“查看代码”，就显示出 Form_订单的代码，如图 6-16 所示。

（2）“查看对象”

其图标为，在“查看代码”图标的旁边，显示选取的工程，可以是文档或是 UserForm 的对象窗口。在单击它之后，出现如图 6-17 所示的窗口图样以及一个工具箱。

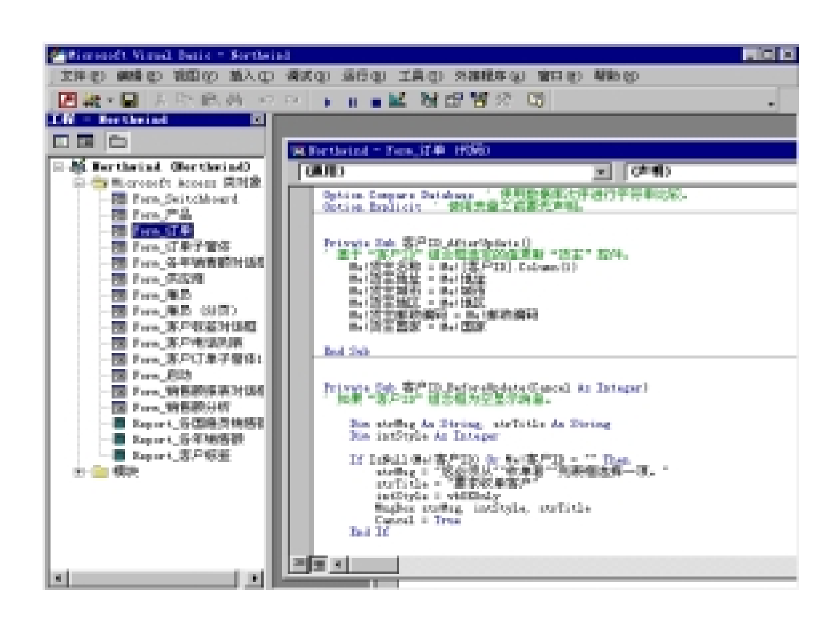


图 6-16　工程资源管理器中按“查看代码”的效果

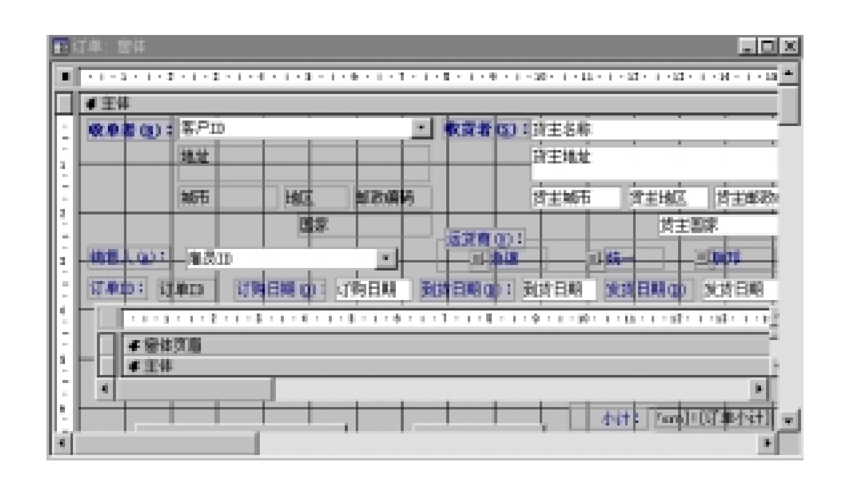


图 6-17　工程资源管理器中按“查看对象”显示窗体的效果

（3）“切换文件夹”

图标是![文件夹图标]，当正在显示包含在对象文件夹中的个别工程时可以隐藏或显示它们。

对比图 6-15 与图 6-18，分别是用户是否按下了“切换文件夹”图标的效果。当按下“切换文件夹”时，如图 6-15 所示，显示了“Microsoft Access 类对象”、“模块”文件夹；而在图 6-18 中，则隐藏了“Microsoft Access 类对象”文件夹、“模块”文件夹，只是将其中的具体内容进行列表。



图 6-18　不按“切换文件夹”的显示效果

（4）列表窗口：

在“查看代码”、“查看对象”和“切换文件夹”的下方的广大区域都是列表窗口。它列出所有已装入的工程以及工程中的工程。

关于工程，图标是 ，如图 6-15 中的 Northwind 就是一个工程。

关于工程以及其包含的工程，又包括以下 6 种形式。下面对照图 6-18 和图 6-16 来加以介绍。

第一种，UserForms 窗体：图标是 ，如图 6-16 所示的“Form_订单”。所谓窗体，是窗口或对话框，窗体为控件的容器，多文档接口

(MDI)窗体可作为子窗体和一些控件的容器。而 UserForms 窗体是所有与此工程有关的.frm 文件。

第二种，Document：是与工程相关的文档。所谓文档，就是任何由一应用程序所创建的工作内容，并给出唯一文件名。例如，在 Microsoft Word 中是 Word 文档。

第三种，模块：图标是，如图 6-18 中“罗斯文帮助函数”等，它是工程中所有的.bas 模块。

第四种，类别模块：图标是，它是工程中所有的.cls 文件。

第五种，ActiveX 设计器：图标是，工程中所有的设计器、.dsr 文件。

第六种，引用：图标是，它实际上是参考其他已经设置使用了 Tools 菜单上的 References 命令的工程。

通过工程资源管理器窗口，用户可以了解工程、工程中的文件、查看其相应的代码，或者对窗体进行操作。

6.2 菜单和快捷键

Visual Basic 有两种类型的菜单：内建方式及快捷方式。

内建菜单出现在 Visual Basic 窗口顶端的菜单栏中，每个菜单名称都会有些相应的命令。举例而言，“格式”菜单包含用来格式化窗体的命令。某些命令具有子菜单，而子菜单又包含一些命令。例如，“视图”

菜单上的“工具栏”命令有一个子菜单，它包含工具条的名称及“自定义”命令。您可以使用“自定义”命令去修改内建菜单或在菜单栏中添加命令。

快捷方式菜单则是一个内含经常使用的命令的菜单，当单击鼠标右键或按 Shift+F10 时就会出现。

关于菜单中命令的信息，还可以使用“帮助”菜单中的“搜索参考索引”命令，然后搜索命令的名称。

在这里，主要介绍以下几种菜单：视图菜单、插入菜单、调试菜单、运行菜单以及文件菜单、工具菜单中的宏命令。

注意：其中一些菜单项并不是在 Visual Basic 编辑器的所有版本中都可用。

6.2.1　视图菜单

视图菜单如图 6-19 所示。

（1）代码窗口命令：按当前所选择的对象，显示或启动过程代码窗口。

工具栏快捷方式：；键盘快捷键：F7。

（2）对象命令：显示活动的项目。

工具栏快捷方式：；键盘快捷键：Shift+F7。

（3）定义命令：显示光标处变量或过程被定义在程序代码窗口中

的位置。如果此定义是一个引用程序库，它会被显示在对象浏览器中。

键盘快捷键：Shift+F2。



图 6-19　视图菜单

（4）最后位置命令：可以让你快速转到上次在过程代码中的位置。

但只能在曾编辑过过程或下过定义命令调用且当前过程代码窗口正显示着时，这个命令才可选用。Visual Basic 只会追踪保留访问过或编辑过的最后 8 行。

键盘快捷键：Ctrl+Shift+F2。

（5）对象浏览器命令：显示对象浏览器，此对象浏览器会列出对象库、类型库、类、方法、属性，事件、及可在过程中使用的常数，和在工程中定义的模块或过程。

工具栏快捷方式：；键盘快捷键：F2。

（6）立即窗口命令：显示立即窗口及其他信息，这些信息可能由

过程代码中的调试语句生成或由直接键入窗口中的命令所生成。

关于立即窗口的作用已经在编辑器部分详细介绍。

工具栏快捷方式：；键盘快捷键：Ctrl+G。

（7）本地窗口命令：显示本地窗口，并且自动显示当前堆栈中所有的变量及其值。

本地窗口会在每次从运行时切换到中断模式或堆栈内容改变时自动地更新。

工具栏快捷方式：。

（8）监视窗口命令：显示监视窗口，并且显示当前的监视表达式。如果该工程定义了监视表达式，则监视窗口会自动出现。

当切换到中断模式时，如果该表达式的内容不在有效范围内，则当前的值不会显示。

工具栏快捷方式：。

（9）调用堆栈命令：显示“调用堆栈”对话框，其中会列出已启动并且尚未完成的过程。只能在中断模式中使用。

当 Visual Basic 在执行一个过程中的程序代码时，该过程会被加入到一个活动过程调用列表中。如果该过程后来又调用了另一个过程，则就会有两个过程在活动过程调用列表中。每当一个过程调用另一个 Sub 过程、Function 过程，或属性过程时，该过程本身就会被加至此列表中。每一个过程会在它返回调用它的过程时从列表中被删除。来

源于“立即”窗口的过程调用也会被加入到此列表中。

也可以单击“本地”窗口中过程对话框旁的“调用”按钮来显示“调用堆栈”对话框。

工具栏快捷方式：，键盘快捷键：Ctrl+L。

（10）工程资源管理器命令：显示工程资源管理器，其中可显示当前打开工程的层次列表及其内容。

工程资源管理器只是一个浏览及管理工具。不能在工程资源管理器中建立应用程序。

工具栏快捷方式：；键盘快捷键：Ctrl+R。

（11）属性窗口命令：显示属性窗口，此属性窗口会依所选择的窗体、控件、类、工程或模块来列出设计时的属性。

工具栏快捷方式：；键盘快捷键：F6。

（12）工具箱命令：显示或隐藏工具箱，此工具箱内包含当前可在应用过程中使用的控件。可以使用附加控件对话框将可插入对象添加到用户工具箱。

工具栏快捷方式：。

（14）工具栏命令：列出所有 Visual Basic 固有的工具栏及自定义命令。可以打开或关闭工具栏或者将工具栏拖至桌面上的另一个位置。如图 6-20 所示。

编辑
标准
调试
用户窗体

自定义(C)...

图 6-20　工具栏所包括的内容

“编辑”显示“编辑”工具栏，此工具栏包含常用的编辑动作的按钮。

“调试”显示“调试”工具栏，此工具栏包含常用的调试动作的按钮。

“标准”显示“标准”工具栏，此工具栏为缺省的工具栏。

“用户窗体”显示“用户窗体”工具栏，此工具栏包含与此窗体有关的按钮。

“自定义”显示“自定义”对话框，可以在此对话框创建或自定义自己的工具栏或菜单栏。

（15）Microsoft Access 命令：将返回 Visual Basic 编辑器的上层—Microsoft Access 的界面。要想重新进入 Visual Basic 编辑器，只要点 Access 的“工具”菜单，弹出“宏”，选其中的“Visual Basic 编辑器”即可。

键盘快捷键：Alt+F11。

以上这些视图菜单的命令已经在第一节中编辑器的各个部分给出

详细介绍，这里就不再举出详细的例图了。

6.2.2　插入菜单

插入菜单包括以下几个部分，如图 6-21 所示。



图 6-21　插入菜单中的内容

（1）插入过程：在当前模块中插入一个新的 Sub 过程、Function 过程或属性过程。非当前模块不能使用。

工具栏快捷方式：。

单击“过程”字样，会出现如图 6-22 所示的“添加过程”对话框。在其中输入要插入的过程名称，再选择该过程的范围、类型，是否定义局部变量为静态变量等，最后点“确定”按钮，就在当前模块中插入了一个相应的过程。

显示在代码窗口的效果如图 6-23 所示。



图 6-22　插入某过程

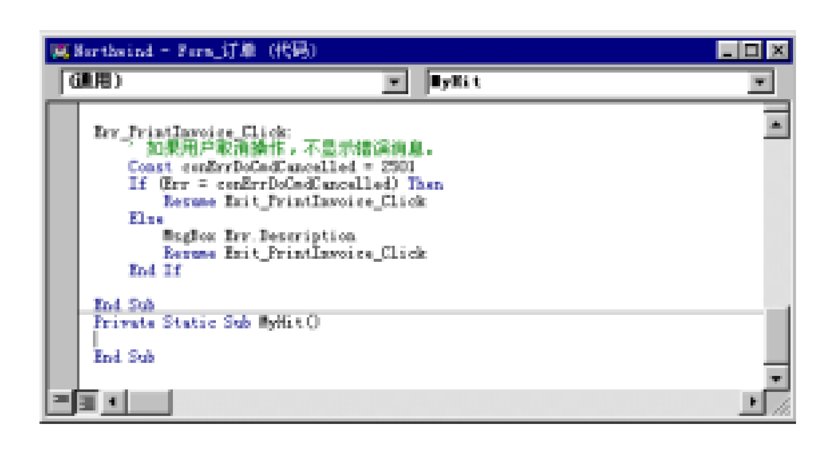


图 6-23　在代码窗口显示了插入某过程

（2）插入模块：将“模块”插入一个新的标准模块并添加到现用工程中。

工具栏快捷方式：。

插入模块的效果如图 6-24 所示。

（3）插入“类模块”命令：创建一个新的类模块并添加到现用工

程。

工具栏快捷方式：。

插入类模块的显示效果如图 6-25 所示。

（4）插入文件：显示“插入文件”对话框，让将原有模块中的文本插入当前光标所在位置。当编辑器未打开时，不能使用。

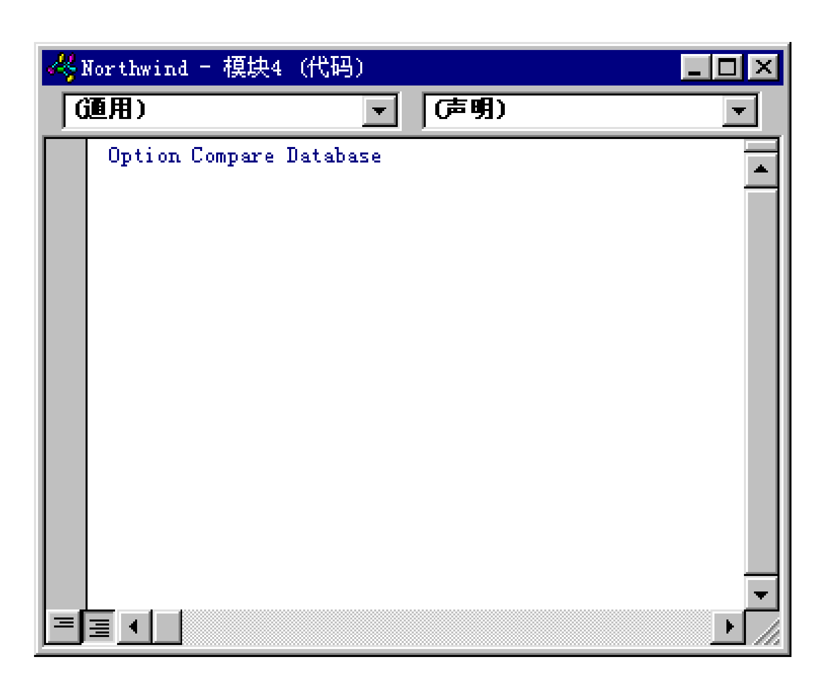


图 6-24　插入一个模块



图 6-25　插入一个类模块

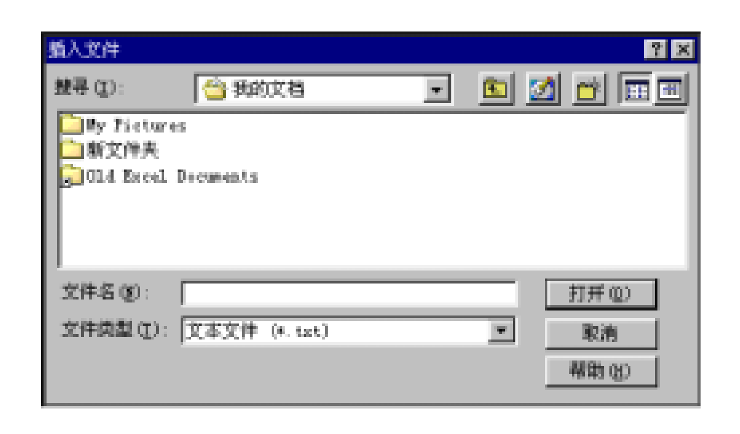


图 6-26　插入文件

6.2.3　调试菜单

在 Microsoft Visual Basic 编辑器中，调试窗口如图 6-27 所示。



图 6-27　调试菜单

（1）编译<Project>：显示如图 6-26 所示的“插入文件”对话框，让将原有模块中的文本插入当前光标所在位置。

（2）逐语句：一次执行一个语句。

当不在设计方式时，“逐语句”会在当前执行行上进入中断模式。如果此语句是对一个过程的调用，下一个被显示的语句就是该过程内的第一个语句。

在设计时，此菜单项会开始执行并在第一行程序被执行前进入中断模式。

如果没有指定当前执行点，“逐语句”命令可能不会有任何作用，除非您以某种方式引发了代码，例如在文档上点一下。

工具栏快捷方式：；键盘快捷键：F8。

如图 6-28 所示，此时显示出“设置下一个语句”的光标，点“逐语句”命令，就每次执行一个语句，黄色光标将跳至下一个可执行语句。如果要取消“逐语句”，就点“跳出”命令。

值得注意的是，此时的“本地窗口”将会显示出该过程出现的所有的变量声明和变量值。可以参阅本章的图 6-11，它列出了图 6-28 过程的变量声明及其值。

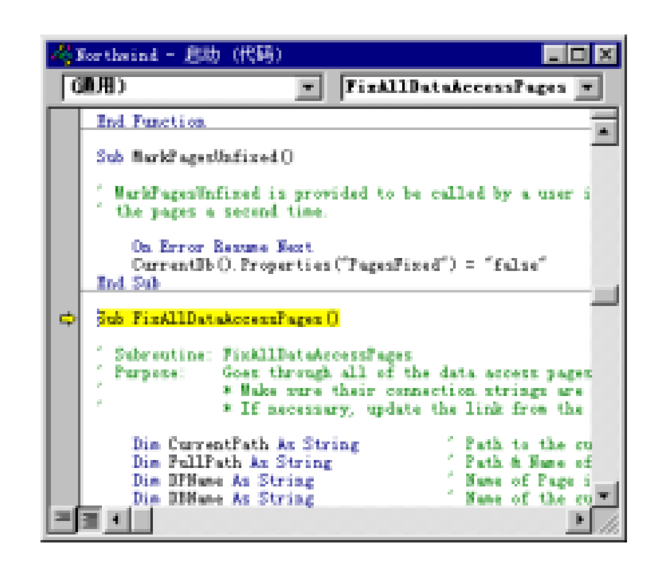


图 6-28　逐语句调试

（3）逐过程：与“逐语句”相似。只有在当前的语句含有一个对过程的调用时，两者才会有差异。

“逐过程”是将过程视为一个基本单位来执行，执行完一个语句后再继续执行下一个语句。不过，下个被显示的语句，就是当前过程中的下一个语句，不会因为当前语句为一过程调用而有所改变。只有在中断模式中可使用。

工具栏快捷方式：；键盘快捷键：Shift+F8。

（4）跳出：执行当前执行点所在函数中剩余未执行的行。下个被显示的语句是紧随在该过程调用后的语句。所有在当前与最后的执行点间的代码都会被执行。此功能仅在中断模式中有效。

工具栏快捷方式：；键盘快捷键：Ctrl+Shift+F8。

（5）运行到光标处：当您的应用程序处于设计方式时，可以使用“运

行到光标处”来选定您想执行到哪一行语句才停止。您的应用程序将会从当前语句执行到您所选定的语句，另外边界标识条会显示在页边距指示区中。

可以使用这个命令来跳过大型循环。

键盘快捷键：Ctrl+F8。

（6）添加监视：显示“添加监视”对话框，用户可以在这个对话框输入一个监视表达式。这个表达式可以是任何的正确的 Visual Basic 表达式。“监视”窗口中的监视表达式会在每次您进入中断模式时更新。

工具栏按钮：。

图 6-29　添加监视

那么，如果在“添加监视”对话框输入某个表达式“Me”，则会在“监视窗口”出现如图 6-30 所示。

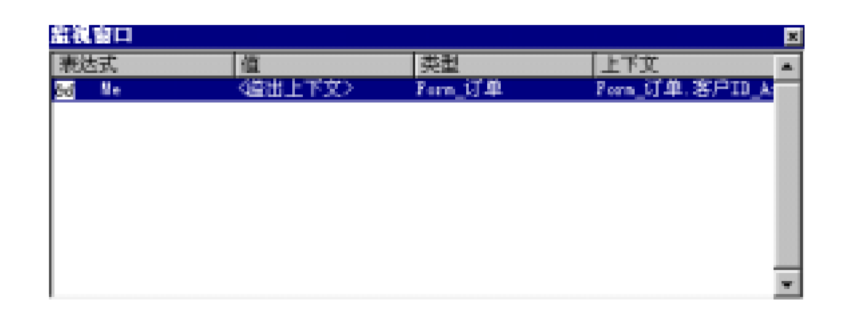


图 6-30　添加对变量“Me”监视后的监视窗口

（7）编辑监视：显示“编辑监视”对话框，用户可以在这个对话框编辑或删除一监视表达式。这个功能可以在有监视时使用，甚至在“监视”窗口处于隐含状态时仍然可以使用。

如图 6-31 所示。

键盘快捷键：Ctrl+W。

图 6-31　编辑监视

（8）快速监视：将选择表达式的值显示在“快速监视”对话框内。如果事先没有为一些变量、属性或其他表达式来定义监视表达式，无

法观察它们的值，这时可以直接使用这个命令来检查其当前的值。

可以从“代码窗口”或是“立即窗口”中选定此表达式，然后选择“快速监视”命令。要想在“快速监视”对话框中添加一个以此表达式为基础的监视表达式，选择“添加”按钮。

工具栏按钮：；键盘快捷键：Shift+F9。

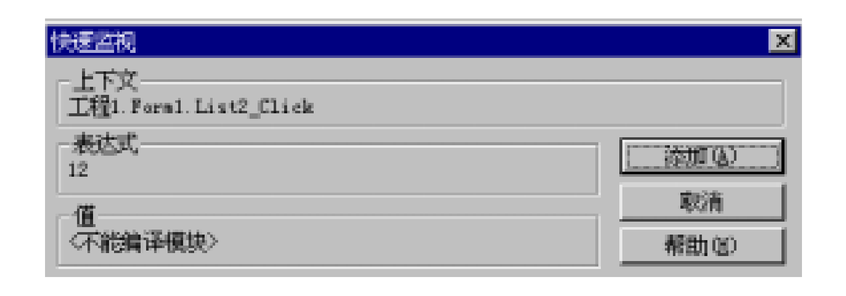


图 6-32　快速监视

（9）切换断点：设置或删除当前行上的一个断点。不能在不含可执行代码的行上（如注释、声明语句或空白行）设置断点。

一行代码的断点颜色可以在“选项”对话框的“编辑器格式”选项卡中指定。

工具栏快捷方式：；键盘快捷键：F9。

如图 6-33 所示。



图 6-33　调试窗口切换断点命令

（10）清除所有断点：清除所有工程中的断点。对于图 6-33 中出现的断点，可以使用该命令使其复原。然而，如果设置了一个监视表达式或是选择了“选项”对话框中“通用”选项卡里的“发生错误则中断”选项，应用程序可能仍然会中断执行。就不能撤消“清除所有断点”命令。

工具栏快捷方式：；键盘快捷键：Ctrl+Shift+F9。

（11）设置下一条语句：将执行点设置到所选择的那行代码。如果想设置一行不同的代码来执行，可以选定一行想执行的代码后选择“设置下一条语句”命令，或是将当前执行行的边界标识条，拖放到想执行那行代码。

使用“设置下一条语句”，可以选择一行位于当前选择语句之前或之

后的代码。当执行代码时，任何位于其中的代码将不会被执行。如果想要重新执行当前过程内的某一语句或是想要跳过一些尚未执行的语句时，便可以使用这个命令。但是不能使用“设置下一条语句”来把执行点设到另一个过程中去。

工具栏快捷方式：；键盘快捷键：Ctrl+F9。

（12）显示下一条语句：突出显示下一个将被执行的语句。使用“显示下一条语句”命令来将光标移到下一行会被执行的语句。

此功能仅可在中断模式中使用。

工具栏快捷方式：。

6.2.4　运行菜单

“运行”菜单如图 6-34 所示。

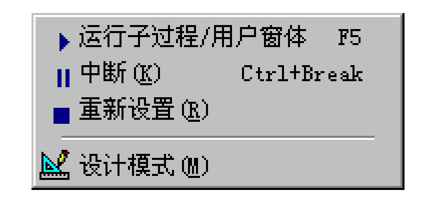


图 6-34　运行菜单中的内容

（1）运行子程序、用户窗体：“运行子程序/用户窗体”执行当前光标所在的过程，或执行当前的窗体。当处在中断模式，这个命令会变成“继续”命令。

如果过程代码窗口或 UserForm 都未被激活，则此处会变成“运行

宏”命令。

当点击此栏时，会产生如图 6-35 的对话框。

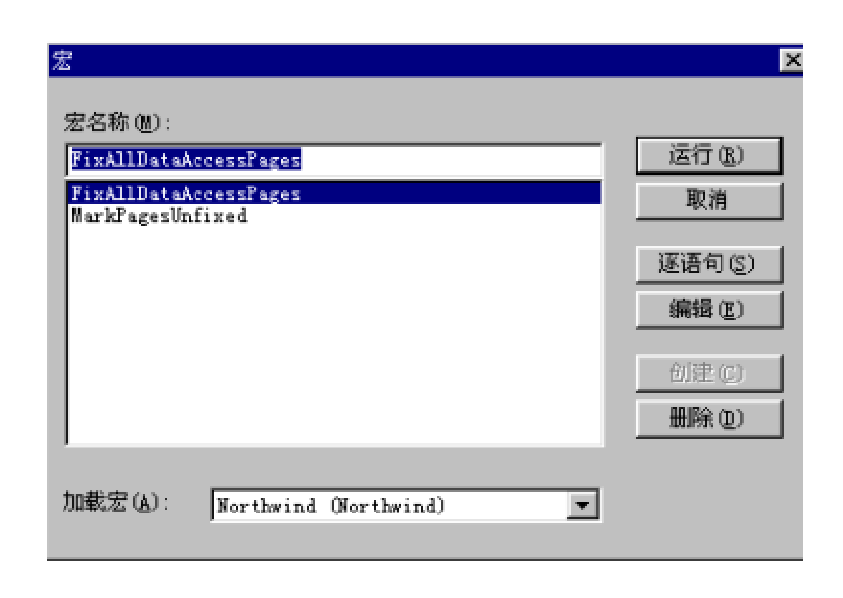


图 6-35 “运行菜单”中的运行宏命令

（2）中断命令：停止一个正在运行的程序，并切换到中断模式。

如果选择“选项”对话框中“编辑格式”选项卡内的“边界标识”条，当选择这个命令之后，则任何正在执行的语句会与出现在带●的“代码”窗口的左边。如果应用程序正在空闲循环中等待事件（没有语句执行），则在事件发生前没有任何语句被执行。

某些在中断方式下所做的更改，必须等到重新启动程序之后，才能生效。

工具栏快捷方式：⏸，键盘快捷键：Ctrl+Break。

（3）重新设置<Project>命令：清除调用堆栈，并清除模块层次的变量。

工具栏快捷方式：。

（4）设计模式与退出设计模式命令：打开或关闭每个工程的设计模式。所谓设计模式是指工程中的过程代码不能执行且来源于主应用程序或工程中的事件也不执行的时候。可以运行宏或用“立即窗口”来离开“设计模式”。

“设计模式”命令是打开设计模式并且改变成“退出设计模式”。

“退出设计模式”命令关闭设计模式并清除工程中所有模块层次的变量。

工具栏快捷方式：。

6.2.5　工具菜单

工具菜单如图 6-36 所示。

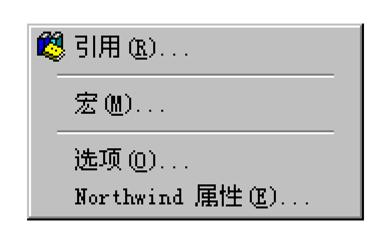


图 6-36　工具菜单的内容

（1）引用：显示“引用”对话框。如图 6-37 所示。

在这个对话框中可添加对象库或类型库引用到工程中，这使过程代码中可使用其他应用过程中的对象。在引用被设定后，此引用对象会显示在“对象浏览器”中。

所谓对象库，就是文件扩展名为.olb 的文件，向自动化控件（比如 Visual Basic）提供有关可用对象的信息。可用对象浏览器检查对象库的内容，以获得有关对象的信息。

所谓类型库，就是其他文件中的文件或部件，包含标准描述，论及用于自动化的已显露的对象、属性、方法。对象库文件(.olb)包含类型库。

也可以将引用添加到其他已装入及存储的工程。如果一个工程尚未存储，会显示为“UNSAVED：<ProjectName>”，则不能引用它。只能应用在设计时。

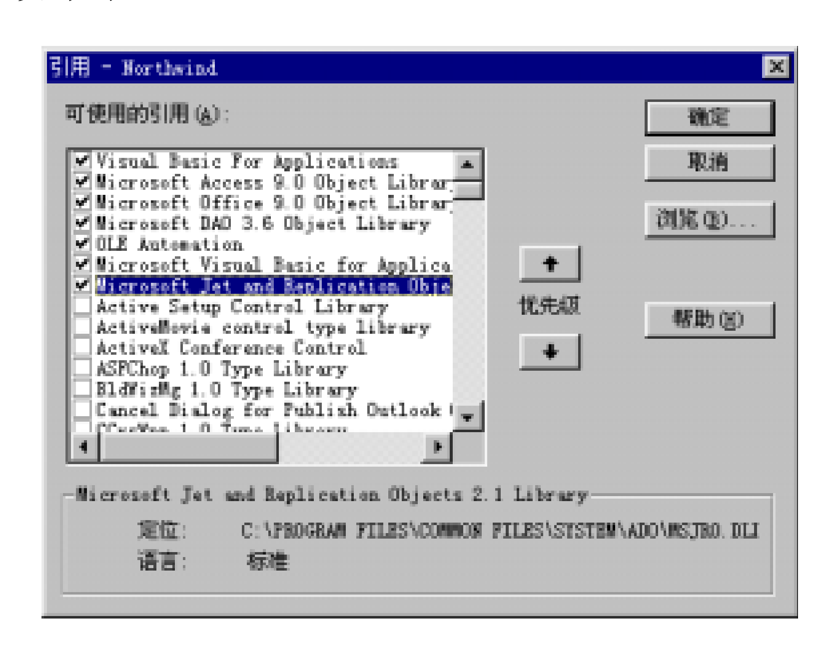


图 6-37　引用对话框

在“引用”对话框中，允许选择另外的应用程序的对象，通过设置对那个应用程序对象库的引用，用户可以在自己的代码中使用它。

对照图 6-37，可以看到对话框的部件主要包括“可使用的引用”、“优先级”按钮、“结果”显示、“浏览”显示四个部分。

“可使用的引用”：列出工程中可引用的工程。

在选定了一对象名称旁的复选框，来设定一个对象程序库的引用之后，就可以从对象浏览中发现此对象及其方法、属性。

如果没有使用到之前引用的对象，必须清除这些工程，可以让 Visual Basic 减少解析的时间，如此可以减少工程的编译时间。

如果删除一个工程正在使用的对象引用，则当下次引用此对象时，将会出现一个错误信息。

按字母顺序来列出非活动的引用。

> 注意：不能删除 Visual Basic for Applications 以及 Visual Basic 对象与过程的引用，因为在执行 Visual Basic 时，它们为必须的。

“优先级”按钮：将列表中设定的引用往上，或往下移动。当代码中引用到某一个对象时，Visual Basic 会按“引用”对话框中的显示顺序来找寻对象。如果两个对象使用相同的名称，Visual Basic 会以较前面的对象为优先。

“结果”：显示“可使用的引用”框中选取对象的名称与路径，以及语言版本，位于引用框的最下部。

“浏览”：显示“添加引用”对话框，可以找寻其他的目录，并将

下列各类型的引用添加到“可使用的引用”框中，如：

类型库(*.olb, *.tlb, *.dll)

可执行文件(*.exe, *.dll)

ActiveX 控件(*.ocx)

所有文件(*.*)

“添加引用”对话框是属于标准的打开对话框。如图 6-38 所示。

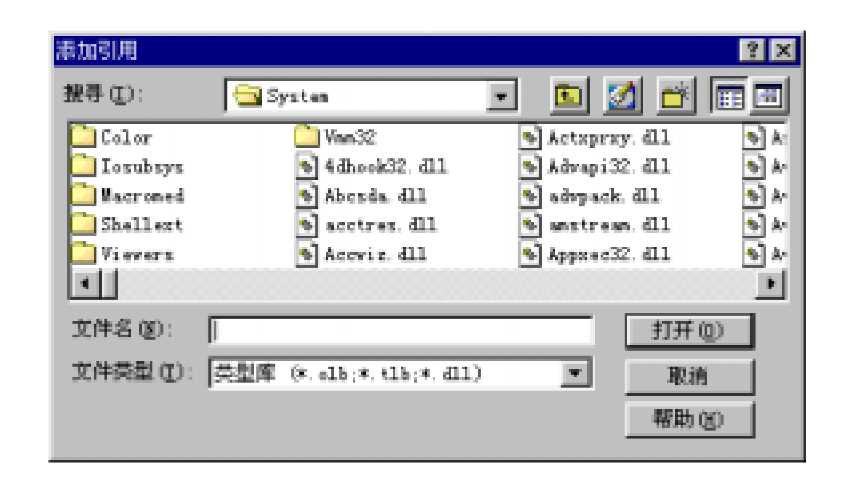


图 6-38　“添加引用”对话框

（2）宏：显示出“宏”对话框，如图 6-35 所示，可以在此“创建”、“运行”、“编辑”或“删除”宏。

在此详细介绍宏对话框中各选项的作用。

宏名：包含选择宏的名称，或“宏框”中没有宏的话则是空白的。

宏框：列出工程中的可用宏。

加载宏：列出可用的工程。

运行：运行选择的宏。

逐语句：突出显示宏的第一行，并放置当前执行行的指示器。

编辑：打开代码窗口并可看见选择的宏，以便修改宏。

创建：在代码窗口中打开一个模块，以创建一个新的宏。

删除：从工程中删除选择的宏。

（3）选项：显示“选项”对话框，如图 6-39 所示。

可以从中选取选项卡，来设置 Visual Basic 编程环境的属性。

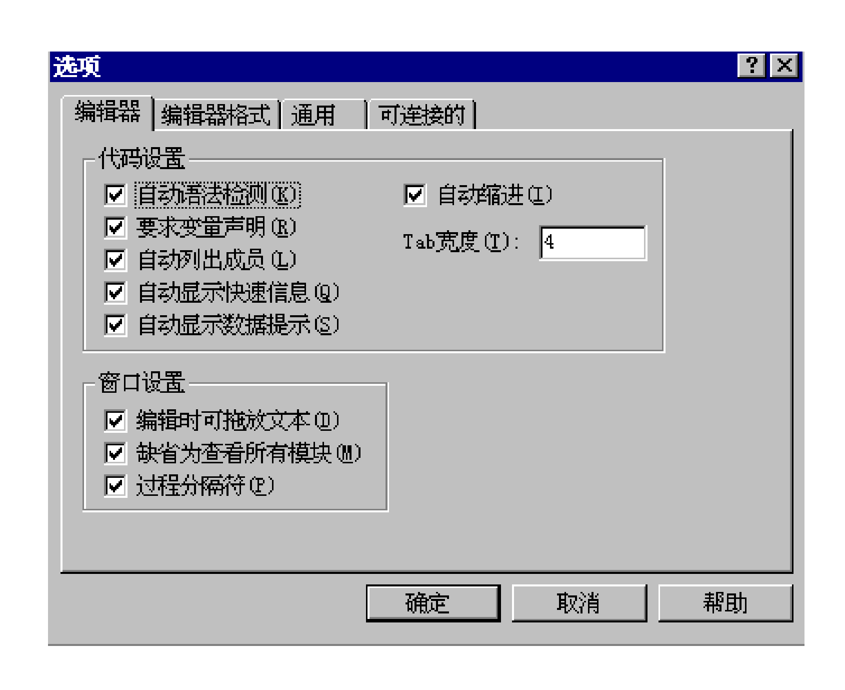


图 6-39　“选项”对话框的“编辑器”

它主要包括“编辑器”、“编辑器格式”、“通用”及“可连接的”四部分。

“编辑器”是对指定“代码”窗口及“项目”窗口的设置。

对照图 6-39，首先是“代码设置”部分，包括自动语法检测（决定 Visual Basic 是否在输入一行代码之后自动校对修正语法），要求变

量声明（决定在模块中是否需要明显的变量声明，选择这个选项会在任一新模块的标准声明中添加Option Explicit语句）、自动列出成员（显示一列表，其包含的信息可以逻辑上的完成当前插入点的语句）、自动快速信息（显示所键入函数及其参数的信息）、自动数据提示（显示出指针所在位置的变量值，只能在中断模式下使用）、自动缩进、Tab 宽度设置定位点宽度。

然后是“窗口设置”部分，包括拖放文本编辑（可以在当前代码中，从“代码”窗口拖放元素到“立即”窗口或“监视”窗口）、缺省为查看所有模块（设置新模块的默认状态，在代码窗口中查看过程，单一滚动列表或是一次只看一个过程。这不会改变当前已打开模块的视图方式）、过程分隔符（可以显示或隐藏“代码”窗口中，出现在每个过程尾端的分隔符条）

图 6-40 显示的是“编辑器格式”。它主要是指定 Visual Basic 代码的外观。这部分包括“代码颜色”、“字体”、“大小”、“边界标识条”、“示例”等内容。

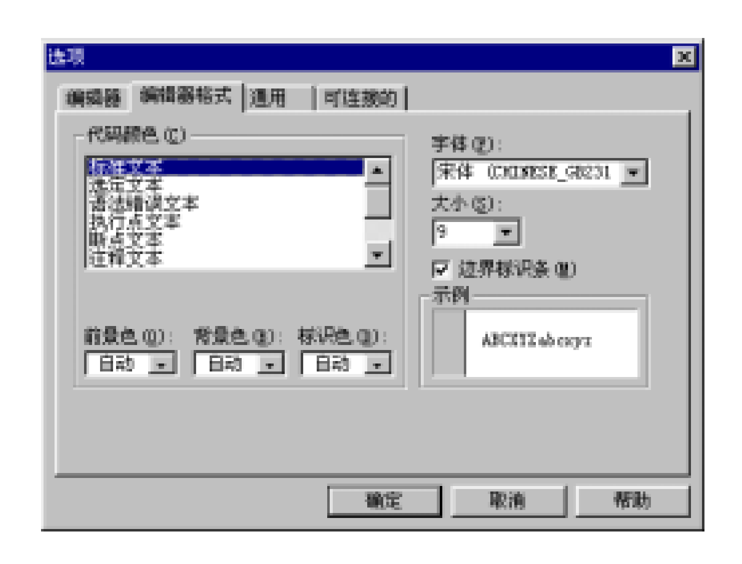


图 6-40　“选项”对话框中的“编辑器格式”

“通用”这一项是指定当前 Visual Basic 工程的设置、错误处理及编译设置，包括“窗体网格设置”（“确定窗体在编辑时的外观”、“显示网格”、“网格单位”、“宽度”、“高度”、“控件对齐网格”）、“显示工具提示”、“折叠 Proj. Hides 窗口”、“编辑并继续”、“错误捕捉”、“编译方式”。

如图 6-41 所示，是选项中的“通用”显示。

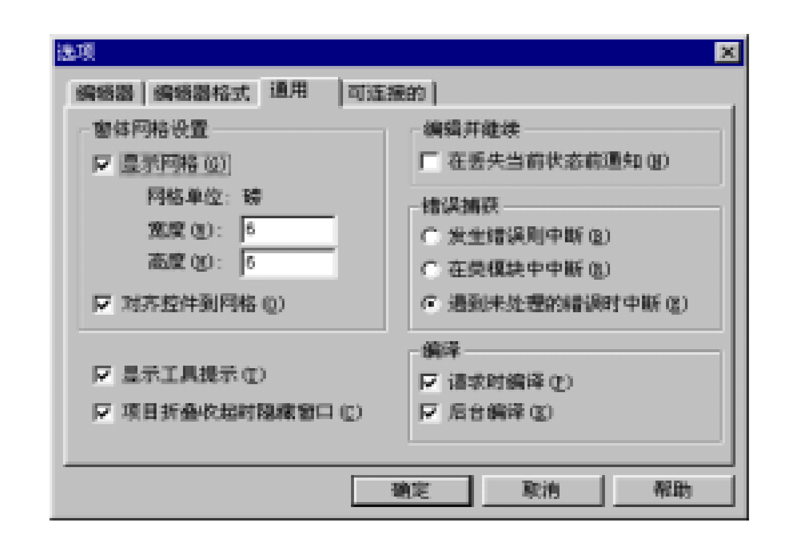


图 6-41　“选项”对话框中的“通用”选项卡

“可连接的”是可以选出要连接的窗口。连接发生在当窗口附加到其他可连接的窗口或应用程序窗口的边缘时。当移动一可连接的窗口时，该窗口很快的移向此位置上。可以移动一个非可连接的窗口到屏幕上的任何地方，并将它留在那里。

选择要使它可连接的窗口，并清除不要的那些。列表中任何、没有或所有的窗口可以连接。如图 6-42 显示的是“可连接的”部分。



图 6-42　“选项”对话框中的“可连接的”部分

（4）工程属性：

显示“工程属性”对话框，对工程所指定的选项设定保护，如图 6-43 所示。



图 6-43 “选项”对话框中的工程属性

图 6-43 实际上显示了 NorthWind 这个工程的工程属性，即对这个工程的保护。

工具栏快捷方式：。

“通用”：指定当前 Visual Basic 工程的设定，工程的名称会显示在标题行中。

包括“工程名称”、“帮助文件名”、“工程说明”、“工程帮助上下文标识符”、“条件编译参数”。

“保护”：设置工程的保护。

其中包括锁定工程（锁住工程并防止其他人改变）、查看工程属性的密码（设置密码，只让某些人可以查看工程的属性）。

6.3　代码的调试

无论多么仔细的生成代码，都可能会出现错误，比如，有时文件会被误删，磁盘驱动空间会溢出，网络驱动器会意外分离等。当发生错误时，处理这些错误，需要将错误处理代码添加到过程之中。

当然，错误也可能出现在代码内部，通常称之为缺陷。例如，光标无法正常指挥，这类的错误就令人无法方便操作。

在应用程序中查找错误并修改的操作叫做调试。Visual Basic 提供了几种调试工具，这些调试工具不但对查出错误根源很有用，而且还能用来尝试改变应用程序，或者用来了解其他应用程序的工作方式。

本节主要内容是说明如何使用 Visaul Basic 的调试工具，以及对代码运行时产生的运行错误进行分析等。

在介绍完本节内容后，应该掌握以下几点：

（1）如何处理错误。

（2）如何设计处理错误的程序。

（3）关闭错误处理。

（4）调试方法以及调试窗口。

（5）避免错误。

（6）设计时、运行时以及中断模式。

6.3.1 错误的类型

开发 VBA 的应用程序的过程主要分为三步：编写程序；调试程序；运行程序。

就理想情况而言，Visual Basic 过程根本不需要进行错误调试。但是，某些硬件的问题或者用户的意外操作都会造成程序错误的产生。这些错误往往会造成代码终止，程序无法运行，或者，使代码产生意想不到的操作。

程序错误也主要分为三类：语法错误、逻辑错误和运行阶段错误。其中，前两类错误是可以避免的，合称为开发错误。

语法错误一般是录入名称错误或者漏写标点符号，或者使用语言函数发生错误。例如，忘记 If…Else…Then 语句要匹配，将某变量名“SomeStr”误写为“SemeStr”等。这类错误比较容易查出。还可以在“工具”菜单中选“选项”命令，在它的对话框内选“编辑器”，单击“自动语法检测”，则会立即显示出错信息。

逻辑错误是当程序运行时没有出错，但产生了非预期的结果的错误。查出错误原因并且改正的难度较大，需要通过使用一系列“调试”代码逐步进行检查。

运行错误发生在应用程序运行的时候。例如，除数表达式为零时，或者要打开不存在的文件时，都会出现这类错误。通过编写错误处理程序，或者编写检查程序环境有效性的代码，可以减少运行程序的错

误。

最好的结果要靠编写容易理解、容易维护的代码，并且拥有有效的调试工具来得到。这项工作需要深入了解过程的工作方式和过程与应用程序的配合方式。

例 1：下面举一个例子说明用代码处理类似无效驱动器或者空软盘的错误问题。

```
Founction  FileExists(filename)  As  Boolean
    Dim  MSg   As  String
    On  Error  GoTo  CheckError
    '如果检测到任何一个错误，则开始捕获。
    Filename=(Dir(filename)<>"")
    '如果未发现错误，则避免使用错误处理程序。
Exit  Founction
    '如果出现错误，就在此分支。
CheckError:
    Const  mnErrDiskNotReady=71,
    _MnErrDeviceUnavailable=68
    '定义常数，表示固定的 Visual Basic 错误代码。
    If  (Err.Number=MnErrDiskNotReady) Then
      Msg="Put  a floppy  diak  in  the  drive"
```

'显示驱动器未准备好的信息。

Msg=Msg&"and close the door"

If MsgBox(Msg,Visual BasicExclamation & Visual BasicOkCancel)=

_Visual BasicOk Then

'显示具有感叹号的图标，及“确定”和“取消”按钮的消息框。

'其中的 Visual BasicExclamation、Visual BasicOk、Visual BasicCancel 是定义在 VBA 库中的常数。

Resume

Else

Resume Next

End If

ElseIf Err.Number= MnErrDeviceUnavailable Then

Msg="This Drive or path does not exist"

'显示装置的路径错误。

Msg=Msg&filename

MsgBox Msg,Visual BasicExclamation

Resume Next

Else

```
            Msg="Unexpected error#"&Str(Err.Number)
            '显示出现预料不到的错误。
            Msg=Msg&"occurred:"&Err.Description
            MsgBox Msg,Visual BasicCritical
            Stop
            '用 Stop 的信号灯图标和“确定”按钮显示消息框。
        End If
        Resume
End Function
```

上面这个例子中，当错误发生时，Visual Basic 将设置错误对象 Err 的各种属性，如错误号，具体描述等等。这样，应用程序就可以智能化的对错误情况作出响应。

例如，当生成“磁盘未准备好”的错误时，代码会显示出一段信息，来通知用户按“确定”或者“取消”按钮。

如果按“确定”，则 Resume 语句将返回到出错语句处，并重新执行该句，这时会出现两种情况。一种情况，如果错误得到改正，操作获得成功；另一种情况，错误仍然存在，程序返回到错误处理程序。

如果用户选择“取消”按钮，则 Resume Next 语句将返回到出错语句之后的那条语句。

当生成装置不可用的错误，代码将显示对该问题进行描述的语句。

当出现预想不到的错误时，就会显示一段描述错误的说明，并在Stop 语句处终止代码。

下面将要详细讨论在程序中修正错误的这些技术。

6.3.2　设计错误的处理程序

对于预感可能出错的任何过程，都要添加错误处理程序。错误处理程序是应用程序中捕获和响应错误的程序。

一般来说，设计一个错误处理程序需要三步：

（1）当错误发生时，通知应用程序在分支点设置错误捕获。例如，例 1 的 On Error GoTo　CheckError 就是一个分支点，它可以激活捕获错误的 CheckError 错误处理程序。

（2）编写错误处理例程，对所有可以预见的错误作出响应。

（3）退出错误处理程序。例 1 中，出现“磁盘未准备好”的错误时，使用 Resume 语句使代码后退到语句出错的地方分支。然后，在出现“装置不可用”的错误时，使用 Resume Next 语句在出错语句之后的那条语句处分支。

下面就分别详细介绍这三个步骤的具体内容。

设置捕获错误：

为设置一个跳转到错误处理程序的错误捕获，可以使用 On Error GoTo line 语句。其中 line 是指出识别错误处理代码的标签。如例 1 中

的函数示例中，标签是 CheckError。

编写错误处理例程：

第一步：添加行标签，标志着错误例程的开始。行标签应该是一个具有描述性的名称，其后必须加冒号。还有这样一个公共约定，把错误处理代码放置在过程结尾，这样，如果未出现错误，过程可以避免执行错误处理代码。

第二步：例程体本身还包括实际处理错误的代码。通常是以 If Then　Else 或者 Case 语句的形式出现，并且还要对每种错误提出相应的操作方法。例如在“磁盘未准备好”的错误下，提示用户插入软盘。

退出错误处理程序：

表 6-2 所示，根据具体情况，可以选用表中的任一语句，实现退出错误处理例程这一操作。

表 6-2　退出错误处理的各语句及意义

语句	意义
Resume[0]	对于出错语句，或者对于最近执行过程调用的语句，程序执行会恢复这些语句。 改正了错误的条件后，可以用它重复操作。
Resume Next	在紧接着出错语句后的那条语句处恢复程序执行。 如果错误发生在错误处理程序之外，而且所调用的程序不具有激活的错误处理程序，则在调用了出现错误的过程后的那条语句，恢复执行。

Resume line	在 line 指定的标签处恢复执行。 此外，line 这个行标签，必须与错误程序在同一过程中。
Err.Raise Number:=number	触发运行时的错误。 在错误处理例程内执行这一语句时，Visual Basic 搜索另一个错误处理例程的调用列表。调用列表是为达到当前执行点而调用的过程链。

这里，有必要简单说明一下 Resume 和 Resume Next 语句之间的差别。

一般来说，错误处理程序无论何时修正错误，都可以使用 Resume 语句；在错误处理程序不能修正错误时，使用 Resume Next 语句。

关于 Resume line 语句的用法，虽然使用它是写代码的合法方法，但是，这种增加行标签的跳转，可能使代码更加难于理解和调试。

6.3.3 关闭错误处理

如果在过程中已经激活了错误捕获，那么，在过程执行完后，错误捕获会自动无效。但是，当过程中的代码一直在执行时，可能会需要关闭过程这的错误捕获。

为关闭错误捕获程序，可以使用 On Error Go To 0 语句，当 Visual Basic 执行该语句时，则在过程内检测错误而不捕获错误。

6.3.4 调试方法和调试窗口

调试方法主要是用来处理前面谈到的三种错误：编译错误、逻辑

错误、运行错误。

Visual Basic 的调试技术包括：断点、中断表达式、监视表达式、通过代码每次经过一个语句或者一个过程、形式变量和属性的值。

此外，Visual Basic 还包括专门的调试功能，例如，可以在过程中进行编辑、设置下一个执行语句、在应用程序处于中断模式时进行过程测试，等等。

下面，分别介绍对于这三类错误的具体的调试方法。

前面已经谈到，编译错误是由于代码书写不正确造成的。例如 For Next 中缺少 Next，那么，Visual Basic 在编译应用程序时会检测出这类错误。这时，使用设置“自动语法检测”选项，避免编译错误。

设置“自动语法检测”的步骤是：先选“工具”菜单中的“选项”，单击该对话框中的“编辑器”选项卡；再选定“自动语法检测”即可。

逻辑错误是当应用程序未按预期方式执行时产生的。从语法角度讲，应用程序的代码是有效的，在运行时也未执行无效操作，但是，还是产生了不正确的结果。在出现了这种情况时，只有通过测试应用程序和分析产生的结果才能检验出来。

运行时的错误是应用程序在运行期间，执行了一个不能执行的操作，就会发生运行错误。例如，有这样一个语句：

Width=Area/Long

如果变量 Long 为 0，除法就是一个无效操作，尽管语句本身的语

法是正确的。当运行应用程序时，才能检测这个错误。

在 Visual Basic 的调试工具中，调试工具栏提供了一些很有用的按钮。这在本章第二节的调试菜单中已有介绍,，其总体效果的快捷键如图 6-44 所示。

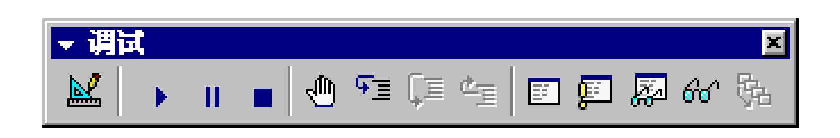


图 6-44　调试工具栏

从左边第四个手标志开始至最右端的工具是调试工具，依次是：切换断点、逐语句、逐过程、跳出、本地窗口、立即窗口、监视窗口、快速监视、调用堆栈。为清楚起见，这里再给出对照列表，如表 6-3 所示。

表 6-3　调试工具及目的

调试工具	目的
断点	在该行终止应用程序的执行
逐语句	执行应用程序代码的下一个可执行语句，并返回到过程。
逐过程	执行应用程序代码的下一个可执行语句，但是并不返回到过程中去。
跳出	执行当前过程的其他部分，并在调用过程的下一行处中断执行。
本地窗口	显示局部变量的当前的值。
立即窗口	当应用程序处于中断模式时，容许执行查询代码或者查询值。
监视窗口	显示选定表达式的值。

快速监视	当应用程序处于中断模式时，列出表达式的值。
调用堆栈	当处于中断模式时，显示一个对话框，呈现所有已经被调用但还未完成运行的过程。

在前面的章节中，已经介绍了如图 6-11 所示的本地窗口，如图 6-10 所示的立即窗口，如图 6-7 所示的监视窗口，如图 6-32 所示的快速监视，没有介绍如下面的图 6-45 所示的调用堆栈对话框。

在“本地窗口”中按“调用堆栈”按钮，就打开了“调用堆栈”对话框。

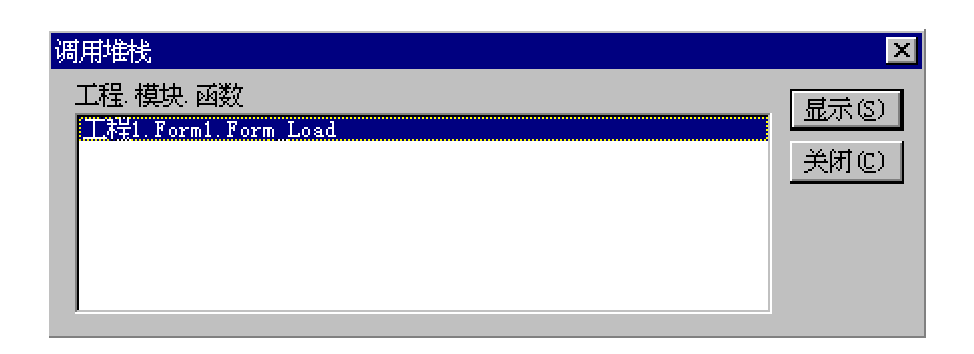


图 6-45　调用堆栈对话框

显示在中断模式期间活动的过程调用。当执行一个过程中的代码时，该过程添加到活动的过程调用的列表。每次过程调用其他过程，便会添加列表。被调用的过程在完成回到原调用过程时便会从列表中删除。从“立即”窗口中调用的过程也会添加调用列表。

注意：仅当应用程序有错时时才需要使用调试工具。

6.3.5　中断模式、运行时及设计时

在测试和调试应用程序代码时，应该知道此时程序所处的模式状况。在“设计时”这种模式下，Visual Basic 创建应用程序；在“运行时”这种模式下，会运行这个程序；在“中断”模式下，会中断程序的执行，利于检查和改变数据。

一般来说，在 Visual Basic 的标题栏，会显示出当前模式。如图 6-46 就分别显示了在“设计时”、“运行时”、“中断模式”三种模式下的标识。

左边是“启动”按钮，中间是“中断”按钮，右边是“结束”按钮。这三个按钮提供了从一种模式切换到另一种模式的作用。



图 6-46　工具栏中的三种模式

这里主要介绍一下如何使用“中断模式”。

在进行调试时，有时需要在认为可能会有问题的地方使代码终止执行。所以，为了这一目的，Visual Basic 提供了断点和 Stop 语句。

工具栏的“中断”按钮已经提供用户主动进入“中断模式”的方法，现在，介绍自动进入中断模式时的几种情况。

（1）语句产生非俘获的运行时错误。

（2）语句产生运行时错误，并且错误捕获选项“发生时中断”

被选中。

（3）“添加监视”对话框中定义的中断表达式发生变化。

（4）执行到一个设有断点的行。

（5）执行到一个 Stop 语句。

✍ 注意：可以在设计时设置断点和监视表达式，但是其他的调试工具只能在中断模式下运行。

当程序处于中断模式时，可以进行以下操作：

（1）在应用程序中修改代码。

（2）观察应用程序界面的情况。

（3）确定哪个活动过程已经被调用。

（4）监视变量值、属性和语句。

（5）改变变量值，改变属性设置值。

（6）查看或者控制应用程序下一步运行的语句。

（7）立即运行 Visual Basic 语句。

（8）手动制作应用程序的操作。

在本章的最后，简单介绍一下简化调试的几种方式：

当应用程序产生的结果不正确时，浏览代码，找出可能产生问题的语法。在这些语句设置断点，并重新启动应用程序。

当程序运行暂停时，测试重要的变量值和变量属性。使用“快速监视”命令或者设置监视表达式来监视这些值。使用“立即窗口”来

检查变量和表达式。

用“发生错误时中断”选项来确定错误发生之处。若要临时改变这个选项，可以从“代码窗口上下文”菜单中选择“切换”，然后，在子菜单中切换选项。

如果错误发生在循环语句，就要定义一个中断表达式，来确定问题出现的地方。在修改代码后，使用“立即窗口”和“设置下一条语句”命令来重新执行这段循环。

如果确定了是变量还是属性导致应用程序的错误，那么，使用 Debug.Assert 语句中断执行这些变量或属性赋予的错误值。

第七章　创建和编辑宏

在本章中，我们将迈出使用 VBA 的第一步。我们将了解到 VBA 与 VISUAL BASIC 以及 VBA 与宏之间的关系，并将通过 Word2000 中的宏录制器录制一个宏，然后使用 Visual Basic 编辑器对宏进行编辑。

宏指令是用 VBA 语言编写的。通过对本章的学习，我们将对 VBA 语言有一个感性的认识。

7.1　什么是 VBA

Visual Basic for Applications(简称 VBA)是新一代标准宏语言。在没有 VBA 以前，一些应用软件如 Excel、Word、Access、Project 等都采用自己的宏语言供用户开发使用，但每种宏语言都是独立的，需要用户专门去学习，它们之间互不兼容，使得应用软件之间不能在程序上互联。一种通用的宏语言可被所有的 Microsoft 可编程应用软件所共享是 Microsoft 公司长期追求的目标。VBA 作为一种新一代的标准宏语言，具有了跨越多种 OFFICE 应用软件并且控制应用软件对象的能力。

Office 2000 包含最新版本的 VBA，它具有和 Visual Basic 语言相

同的功能。VBA 为 Office 2000 提供了多种功能，例如无模式用户窗体及对附加 ActiveX 控件的支持。

从 VBA 的定义即可看出，它跟宏和 VISUAL BASIC 有着密不可分的联系：

7.1.1　VBA 与宏的关系

VBA 与宏有着密切的联系，但是它们之间不能划上等号。宏是一串指令（比如说：Word 命令）构成的集合，它的作用是让 OFFICE 程序执行一连串的操作。而 VBA 是一种面向对象的宏语言，它具有一系列高级语言才具有的特性。并且 VBA 的功能更强大，它能够通过对对象（包括可见对象和不可见对象）来实现宏不能实现的功能。

当我们记录一个宏时，Word 将我们的操作记录为一个的 VBA 过程，我们可以通过 VISUAL BASICE（VISUAL BASIC 编辑器）来编辑宏，增添一些不能记录的功能。

7.1.2　VBA 与 VISUAL BASIC 的关系

VBA 和 VISUAL BASIC 都是一种面向对象的，可视化的，事件驱动的高级语言。VBA 是在 VISUAL BASIC 的框架上建立起来的，它的语法、功能甚至开发环境跟 VISUAL BASIC 都是基本相同的。在最新推出的 OFFICE 2000 中包含的 VBA 6.0(亦即 VBA 2000)则是 Visual Basic 6.0 的子集。

不同的是，VBA 专门用于 Office 的各应用程序。VISUAL BASIC 可运行直接来自 Windows 95 或 NT 桌面上的应用程序，而 VBA 的项目(Project)仅由使用 VBA 的 Excel、Word、PowerPoint 等称为宿主(Host)的 Office 应用程序(Application)来调用。

通过上面的描述，我们可以看到 VBA 有以下特征：

- VBA 是一种解释性语言

  由于 VBA 是在 VISUAL BASIC 的框架上建立起来的，因此它也继承了 VISUAL BASIC 作为一种解释性语言的特性。

- VBA 是一种面向对象的语言

  作为新一代的高级语言，VBA 提供了对面向对象的程序设计方法的支持。实际上，VBA 的工作过程就是对各种对象（包括 VISUAL BASIC 对象和 OFFICE 对象）进行操作的过程。

- VBA 支持可视化的编程环境

  VBA 提供了新颖的可视化设计工具，巧妙地将 Windows 界面设计的复杂性封装起来，程序开发人员不必再为界面设计而编写大量程序代码，仅需采用现有工具按设计者要求的布局，在屏幕上画出所需界面，并为各图形对象设置属性即可

- VBA 不能单独被执行，只能被 OFFICE 程序所调用。
- VBA 是事件驱动的。

7.2　录制第一个宏

我们将录制一个宏，用于改变选定文字的格式，样式，字体，颜色和方向。当然，这样的一个宏是很少会用到的。但在这里，我们只是要举一个简单的宏的例子。在日常的工作中，我们必须考虑到录制一个宏的“经济”性，通常我们只将最常用的一串（或者一个）指令录制为宏。

宏的录制步骤如下：

（1）选定需要改变格式的文本。

（2）选择“工具”菜单中的“宏”子菜单中的“录制新宏”命令。打开的“录制新宏”对话框如图 7-1 所示。

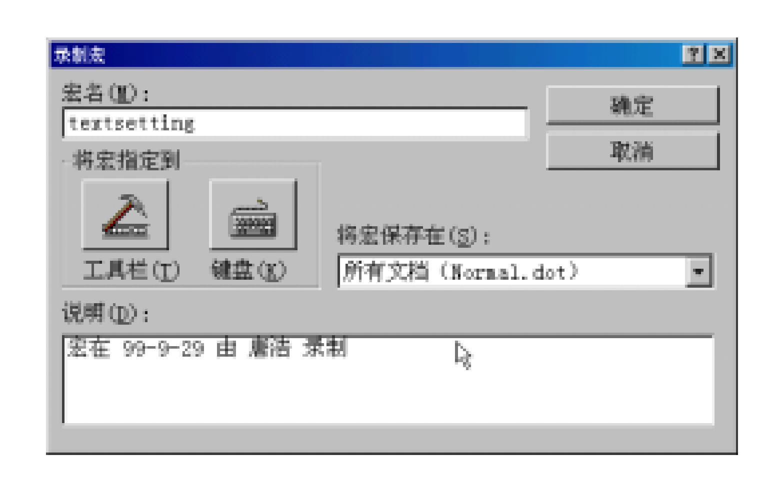


图 7-1　“录制宏“对话框

（3）在“宏名”文本框中输入宏的名字，在这里我们输入“textsetting”作为宏名。

（4）在“说明”文本框中输入对宏的说明。

（5）单击“确定”按钮，开始宏的录制。

（6）将选定文本的样式设置为“标题 3，格式设置为“Times New Roman”，字体大小设置为“小初”、“斜体”，将字体颜色设置为“红色”。

（7）单击“停止录制”按钮，停止宏的录制。

如果录制的宏要常常用到的话，可以将其指定到工具栏中或者给它指定一个快捷键。具体的方法可见前面的章节。

从 Word 中录制宏的确非常方便。但是在录制的宏的功能会受到一些限制,例如不能记录下列宏：

- 条件分支
- 变量指定
- 循环结构
- 自定义用户窗体
- 错误控点
- 用鼠标选定的文字（必须使用组合键）

要提高所录制宏的功能，可能需要修改录制到模块中的代码。

7.3　使用 Visual Basic 编辑器编辑宏

我们可以 Visual Basic 编辑器来编辑已经创建的宏，当然我们还可以用它来从头创建一个宏，其步骤如下：

（1）选择“工具”菜单中、“宏”子菜单中的“宏”命令，打开的“宏”对话框如图 7-2 所示。

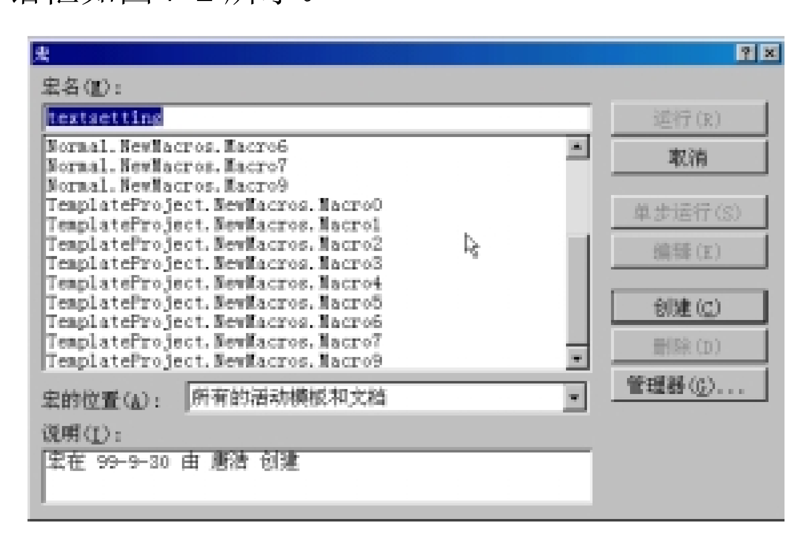


图 7-2 “宏“对话框

（2）在列表中选择要编辑的宏，如我们的“textsetting”宏，如果要选择的宏不在列表中，我们应先在“宏的位置”下拉框中选择宏所在的正确位置，然后在列表中选择宏。

（3）单击“编辑”按钮，打开 Visual Basic 编辑器（如图 7-3 所示）对宏进行编辑。

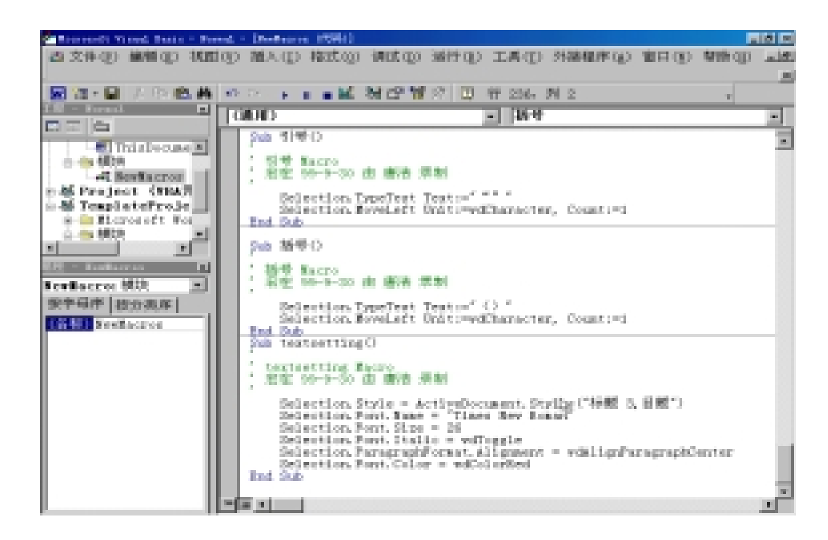


图 7-3　Visual Basic 编辑器

在 Visual Basic 编辑器的代码窗口中，我们可以看到“textsetting”宏的代码如下：

“textsetting”宏代码

1．　Sub textsetting()

2　'

3　' textsetting Macro

4　' 宏在 99-9-30 由 唐浩 录制

5　'

6　Selection.Style = ActiveDocument.Styles("标题 3,目题")

7　Selection.Font.Name = "Times New Roman"

8　Selection.Font.Size = 26

9　Selection.Font.Italic = wdToggle

```
10 Selection.Font.Color = wdColorRed
11 Selection.ParagraphFormat.Alignment = wdAlignParagraphCenter
12 End Sub
```

我们会发现，宏代码实际上是 VISUAL BASIC（VBA）代码。

下面就让我们来了解各行宏代码有什么意义。为了方便起见，我们把每行代码都加上了行号。

第一行：宏代码的开始。可见，一个宏在 VISUAL BASIC 代码中实际上是一个过程，宏中的各指令则是这个过程中的可执行语句。

第二到第五行：对宏代码的注释。Word 会自动给宏代码加上注释，注释的内容跟我们在创建宏时填写的宏的说明是相同的。

第六行：将选定文本的样式变为“标题 3，目题”。

第七行：将选定文本的字体变为“Times New Roman”。

第八行：将选定文本的字号大小改为 26 磅。

第九行：将选定文本改为斜体。

第十行：将选定文本改为红色。

第十一行：将选定文本所在段落的对齐方式改为“居中对齐”。

第十二行：结束宏操作。

我们在前面讲过，录制的宏有一些功能限制。但是我们可以通过

对用 Visual Basic 编辑器对宏进行编辑来扩充宏的功能。

在上面的例子中，我们可以增添“textsetting”宏的功能，使我们运行完“textsetting”宏后，出现一个对话框表明宏操作已经完成。我们只需要在第 11 行和第 12 行之间插入如下一条语句即可：

box = MsgBox("Text changed", Visual BasicOKOnly)

保存对宏代码所作的修改，按下“Alt＋F11”键切换到 Word 界面。选定文本后，从宏对话框中运行“textsetting”宏。在对文本的格式操作完成以后，Word 会打开一个对话框（如图 7-4 所示）表明宏已经运行完毕。



图 7-4　运行“textsetting”宏打开的对话框

也许您还看不懂上面的各行代码是什么意思，这并没有什么关系，在后面的章节中，我们将 VBA 程序的熟悉程度将会逐步加深。在本章中，我们只要了解到下面几点就可以了：

（1）我们在录制宏时，实际上是将对 Word 的操作转换为 VBA 代码。在执行宏时，这些代码将被调用。

（2）对于 VBA 代码而言，每条宏命令都是一个过程，而宏命令中的各步操作则是过程中的可执行语句。

（3）我们可以使用 Visual Basic 编辑器对录制的宏进行修改。使宏的功能更加强大。

第八章　Word2000 的对象模型

在前面我们已经了解到，VBA 是一种面向对象的语言，它的语法和 VISUAL BASIC 是基本相同的。跟 VISUAL BASIC 相比，它甚至能作更多的事，因为 VBA 还能够对 OFFICE 对象（包括 Word 对象）进行操作。对象是代码和数据的组合，可以作为一个单位来处理。对象可以是应用程序的一部分，比如可以是控件或窗体。整个应用程序也是一个对象。

对象是 Visual Basic 这座大厦的基石，几乎在 Visual Basic 中的每个操作都与修改对象有关。Word 的任何元素（如文档、表格、段落、域、书签等）都可以用 Visual Basic 中的对象来代表。同样，Word 对象代表一个 Word 的元素，如文档、段落、书签或单个的字符。

要想掌握 VBA 在 Word 中的应用，了解 Word 的对象模型是必不可少的。

Word2000 的对象模型是一个有层次的对象的集合，这里的层次指的是对象之间的组织结构，例如，两个对象之间是平行关系，还是从属关系。Word2000 对象模型的层次结构决定了对象间的相互关系以及访问它们的方法。

Word2000 的对象模型如下图所示。



图 8-1　Word 的模型结构图

从图 8-1 中可见，Word2000 中的最高级对象是 Application 对象，其他所有的 Word2000 对象都是它的子对象。下面，我们将以 Application 对象为起始点，对 Word2000 的对象模型进行介绍。在本章中将介绍最常用的 Word 对象和它们的一些应用，以及一些相关的知识。

8.1　Application 对象

在 Word2000 中，我们能够获得的最高层对象是 Application 对象，它代表的是 Word2000 应用程序本身。在 Application 对象中包含了一些其他的对象（集合），例如 Word2000 的 Application 对象中包含了

Document、windows、Selection 等对象。

如要在其他应用程序中通过自动对象（即以前的 OLE 自动功能）控制 Word 时，则可用 **CreateObject** 或 **GetObject** 函数返回一个 Word Application 对象。下列 Microsoft Excel 示例启动 Word（如果尚未启动），然后打开一篇原有文档。

```
Set wrd = GetObject(, "Word.Application")
wrd.Visible = True
wrd.Documents.Open "C:\My Documents\Temp.doc"
Set wrd = Nothing
```

下面将讲述 Application 的常用属性，事件和方法。

8.1.1 Application 对象的属性

1. Application 属性

用 Application 属性可返回一个 Application 对象。

下列示例显示使用的 Microsoft Word 的版本号。

```
MsgBox Application.Version
```

（1）打开宏对话框（Alt＋F8），选择创建一个宏，宏名任意。将上面的语句写入宏中，再运行该宏，我们将会看到如图 8-2 所示的对话框。



图 8-2　用对话框显示 Word 的版本号

- 我们使用的是 Microsoft Word2000，其版本号为 8.0。
- “全局属性和方法”

  尽管 Application 对象是 Word 对象中最高层的对象。但是这并不意味着我们在使用任何 Word 对象时都要通过 Application 对象来区分。在使用可返回最普通的用户界面对象的许多属性和方法（例如活动文档，即 ActiveDocument 属性）时，可不必用 Application 对象进行区分。例如，我们可以直接使用 ActiveDocument.Visiable，而不必写为 Application.ActiveDocument.Visiable。
- 使用时可以直接调用，不受 Application 对象限制，属性和方法被看作是“全局”的。要查看一个属性（方法）是不是全局属性（方法），可以按如下的步骤进行:

  1. 打开 Visual Basic 编辑器，其快捷键为 Alt + F11。

  2. 打开“视图菜单”（View），选择“对象浏览器”命令。打开的对象浏览器如图 8-3 所示。

  3. 在“工程，库”框中（最左上方的下拉选择框）选择“Word”。

4. 在“类别”框中选择列表顶端的“〈全局〉”项，即可在“对象浏览器”中查看 Word 对象的共用属性和方法。

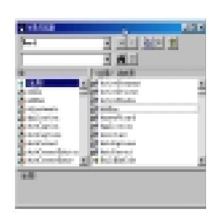


图 8-3　用“对象浏览器”查看全局属性和方法

1. ActiveDocument 属性

activewindow 是 application 对象的一个只读属性，它返回一个 **Document** 对象，该对象代表活动文档。如果没有打开的文档，就会产生错误。

本示例保存当前的活动文档，如果没有打开的文档，则新建一个文档。

```
If Documents.Count >= 1 Then
    ActiveDocument.Save
Else
    Documents.Add
End If
```

2. ActiveWindow 属性

ActiveWindow 属性也是一个只读属性，它返回一个 **Window** 对象，该对象代表当前的活动窗口。如果没有打开的窗口，则会产生错误。

下面的示例为当前的活动文档打开一个新窗口，然后平铺所有Word 程序窗口。

```
Set myWindow = ActiveDocument.ActiveWindow.NewWindow
Windows.Arrange ArrangeStyle:=wdTiled
```

3. assistant 属性

assistant 属性是一个只读属性，用它可以返回一个 **Assistant** 对象，该对象代表 Office2000 的助手。

下面的示例将会在指定的地方打开 Office 助手，并且让它动个不停。

```
With Assistant
    With Assistant
      .On = True
      .Visible = True
      .Move xLeft:=400, yTop:=300
      .Animation = msoAnimationCheckingSomething
End With
```

4. comandbars 属性

comandbars 属性为只读属性，用它可以返回一个 **CommandBars** 集合，该集合代表 Word 的菜单栏及所有工具栏。

下面的示例在应用程序窗口的底部显示“绘图”工具栏。

```
With CommandBars("Drawing")
  .Visible = True
  .Position = msoBarBottom
End With
```

5. documents 属性

用 documents 属性返回一个 **Documents** 集合，该集合代表所有打开的文档。它也是一个只读属性。

下面的示例将会保存当前文档之后将文档关闭。

```
      With ActiveDocument
   If  .Saved=false
      Then  .Save
            .Close
      End If
   End With
```

6. tasks 属性

只读属性，返回一个 **Tasks** 对象集合，它代表了所有正在运行的应用程序。

下面的示例显示计算器。如果计算器还没有运行，则 Word 启动该任务，然后显示计算器。

```
If Tasks.Exists("Calculator") Then
```

```
  With Tasks("Calculator")
  .Activate
  .WindowState = wdWindowStateNormal
  End With
Else
  Shell "calc.exe"
  Tasks("Calculator").WindowState = wdWindowStateNormal
End If
```

下面的示例查看 Microsoft Excel 是否正在运行。如果该任务正在运行，本示例将激活 Microsoft Excel；否则，将显示一个消息框。

```
If Tasks.Exists("Microsoft Excel") = True Then
  With Tasks("Microsoft Excel")
  .Activate
  .WindowState = wdWindowStateMaximize
  End With
Else
  Msgbox "Microsoft Excel is not currently running."
End If
```

7. Windows 属性

只读属性，用 Windows 属性可以返回一个 **Windows** 集合。Windows

属性可以被 Application 对象和 Document 对象调用。

当 Windows 属性被 Application 对象调用时，它返回的 Windows 集合表示所有的文档窗口，集合中的各个对象一一对应于“窗口”菜单底部显示的窗口名。

当 Windows 属性被 Document 对象调用时：返回的 **Windows** 集合表示指定文档的所有窗口（例如，Example.doc:1 和 Example.doc:2）。

下面示例显示 NewWindow 方法（创建一个新窗口）应用前后活动文档的窗口数目。

```
MsgBox Prompt:= ActiveDocument.Windows.Count & " window(s)", _
    Title:= ActiveDocument.Name
ActiveDocument.ActiveWindow.NewWindow
MsgBox Prompt:= ActiveDocument.Windows.Count & " windows", _
    Title:= ActiveDocument.Name
```

下面的示例分别显示所有文档的窗口数目和当前文档的窗口数目。

```
MsgBox Prompt:=Application.Windows.Count,_
    Title:="windows number of all documents "
MsgBox Prompt:=Activedocument.Windows.Count,_
```

Title:="windows number of active document "

8. WindowState 属性

WindowState 属性是一个可读写的属性，它返回或设置指定的文档窗口或任务窗口的状态。可选下列 WdWindowState 常量之一：wdWindowStateMaximize、wdWindowStateMinimize 或 wdWindowStateNormal。这些常量都是 Long 类型。

应当注意的是，wdWindowStateNormal 常量指明窗口没有最大化或最小化。不能设置非活动窗口的状态。在设置窗口状态之前可用 **Activate** 方法激活窗口

下面的示例将会最小化 Microsoft Excel 应用窗口。

```
For Each myTask In Tasks
  If InStr(myTask.Name, "Microsoft Excel") > 0 Then
  myTask.Activate
  myTask.WindowState = wdWindowStateMinimize
  End If
Next myTask
```

8.1.2 Application 对象的方法

1. activate 方法

激活指定的 Window 或 Document 对象,它有两种调用方式。

语法 1：

object.Activate(Wait)

object 不可省略，它表示要被激活的 Task 对象。

Wait 为 Boolean 类型，只适用于 Task 对象。如果本参数为 True，则只在用户激活 Word 之后才会激活任务。如果本参数为 False，则即使 Word 不处于活动状态，也可以立即激活该任务。

整个表达式返回一个 Task 对象。

在下面的示例中，如果如果 Word 处于活动状态，且已运行 Excel 程序，则激活 Excel 程序。

Tasks（"Microsoft Excel"）.activate（true）

语法 2：

object.Activate

object 不可省略，它表示要被激活的对象。整个表达式返回 object 对应的对象。

下面的示例激活 Windows 集合中的下一个窗口。

ActiveDocument.ActiveWindow.Next.Activate

在下面的示例中，如果“记事本”应用程序在 Tasks 集合中，则激活“记事本”。

```
For Each myTask In Tasks
  If InStr(myTask.Name, "Notepad") > 0 Then
```

```
myTask.Activate
myTask.WindowState = wdWindowStateNormal
End If
Next myTask
```

2. NewWindow 方法

Application 对象和 Window 对象都可调用 NewWindow 方法。调用 NewWindow 方法将打开一个新的窗口作为指定窗口，并显示原文档。然后返回一个 Application 或 Window 对象。

语法

object.NewWindow

该表达式返回一个 Application 或 Window 对象。

如果将 NewWindow 方法应用于 Application 对象，则为活动窗口打开一个新的窗口。对同一文档打开多个窗口时，窗口标题将出现冒号（：）和一个数字。例如我们为 Example.doc 打开两个窗口，则第一个窗口的标题为“Example.doc：1”，第二个窗口的标题为“Example.doc：2”。

下列两条指令在功能上是等效的。

```
Set myWindow = ActiveDocument.ActiveWindow.NewWindow
Set myWindow = NewWindow
```

下面的示例将为当前的活动文档打开一个新窗口，并依次显示各

个窗口的标题。

```
ActiveDocument.ActiveWindow.NewWindow
For Each mywindow In Windows
MsgBox mywindow.Caption
Next
```

3. Quit 方法

Quit 方法的作用是退出 Word，并可选择保存或传送打开的文档。

语法

object.Quit(SaveChanges, Format, RouteDocument)

表达式返回一个 Application 对象。

object Application 对象，不可省略。

SaveChanges：Variant 类型，可选。指定退出 Word 前是否保存修改过的文档。可以是下列 WdSaveOptions 常量中的任何一个：

wdDotNotSaveChanges 退出，且不保存文档

wdPromptToSaveChanges 退出，并显示“保存文档”对话框

wdSaveChanges。 退出，并保存文档

OriginalFormat：Variant 类型，可选。指定 Word 对原为非 Word 格式的文档的保存方式。可以是下列 WdOriginalFormat 常量中的任何一个：

wdOriginalDocumentFormat　　将文档保存为原先的格式

wdPromptUser　　显示“保存格式”对话框

wdWordDocument。　　将文档保存为 Word 文档

RouteDocument：Variant 类型，可选。如果属性值为 True，则发送文档给下一个收件人。如果文档没有附属的发送名单，则该参数无效。

下例将所有文档保存为 Word 格式，并退出 Word。

Application.Quit SaveChanges:=wdSaveChanges, OriginalFormat:=wdWordDocument

在上面我们介绍了 Application 对象的常用属性和方法，由于 Application 对象是我们讲述的第一个 Word 对象，所以我们对它的常用属性和方法进行了详尽的介绍，并且给出了大量的例子，我们希望通过这样能够使读者对调用 Word 对象的属性和方法有一个初步的认识，在下面我们还将讲述一些 Word2000 对象模型的常用对象，但我们不会再对各个对象的属性和方法进行一一的介绍。我们将教给读者如何查看对象的属性和方法。

我们可以通过下面两种方法来查看一个对象的属性和方法。

- 通过 Office 2000 的在线帮助查看对象的属性和方法

通过 Office 2000 的在线帮助查看对象的属性和方法的步骤如下：

（1）打开 Office 2000 的在线帮助

（2）在“目录”栏中选择“与编程有关的信息”、“Microsoft Word Visual Basic 参考”、“Microsoft Word 对象”。

（3）从右边的对象结构图中选择要查看的对象

（4）点击“属性”或“方法”即可查看对象的属性和方法。

- 通过“对象浏览器”查看对象的属性和方法

（1）打开 Visual Basic 编辑器。

（2）打开“视图”菜单中的“对象浏览器”命令，打开对象浏览器。

（3）如果要查看Word对象的话，在“工程/库”选项框中选择“Word”库。如果我们要查看其他库的对象，则可以选择其他的库。

（4）在“类”列表框中选择要查看的对象

（5）此时即可在“成员”列表框中查看对象的属性和方法。

在上面我们讲述了 Application 对象的属性和方法，下面我们将讲述 Appliation 对象的事件以及如何使用 Application 对象的事件。

按照下面三个步骤可生成 Application 对象事件句柄：

- 在类模块中声明对应于事件的对象变量。
- 编写指定事件的过程。
- 从其他模块中初始化已声明的对象。

（1）声明对象变量

在为 Application 对象事件编写过程之前，必须新建一个类模块并

声明一个包含事件的 Application 类型对象。例如，假设已建立新类模块并将其定名为 EventClassModule。该类模块包含以下代码:

```
Public WithEvents App As Word.Application
```

（2）编写事件过程

定义了包含事件的新对象后，它将出现在类模块的“对象”下拉列表框中，然后可为新对象编写事件过程。（在“对象”框中选定新对象后，其有效事件将出现在“过程”下拉列表框中）。从“过程”下拉列表框中选择一个事件，则在类模块中会增加一空过程。

```
Private Sub App_DocumentChange()
…
End Sub
```

（3）初始化已声明对象

在过程运行之前，您必须将类模块中的已声名对象（本例中为 App）和 Application 对象相连接。您可在任何模块中使用以下代码：

```
Dim X As New EventClassModule
Sub Register_Event_Handler()
  Set X.App = Word.Application
End Sub
```

8.1.3 Application 对象的事件

1. DocumentBeforeClose 事件

DocumentBeforeClose 事件在打开的文档关闭之前发生。

语法：

Private Sub object_DocumentBeforeClose(ByVal Doc As Document, ByVal Cancel As Boolean)

object 为 在类模块的事件中声明的 Application 类型对象。

Doc 为要关闭的文档。

Cancel：如果为 False，事件发生。如果事件过程将该参数设为 True，则过程完成后文档不会关闭。

本示例在文档关闭前，提示用户用“是”或“否”进行响应。

```
Private Sub App_DocumentBeforeClose _
 (ByVal Doc As Document, _
 Cancel As Boolean)
 a = MsgBox("Do you really " _
 & "want to close the document?", _
 Visual BasicYesNo)
 If a = Visual BasicNo Then Cancel = True
End Sub
```

2. DocumentBeforePrint 事件

该事件在打开的文档打印之前发生。

语法：

Private Sub object_DocumentBeforePrint(ByVal Doc As Document, ByVal Cancel As Boolean)

Object：在类模块的事件中声明的 Application 类型对象。

Doc：将打印的文档。

Cancel：如果为 False，则事件发生。如果事件过程将此参数设为 True，则过程完成后将不打印文档。

3. DocumentBeforeSave 事件

该事件在打开的文档保存之前发生。

语法：

Private Sub object_DocumentBeforeSave(ByVal Doc As Document, ByVal SaveAsUI As Boolean, ByVal Cancel As Boolean)

Object：在类模块的事件中声明的 Application 类型对象。

Doc：将保存的文档。

SaveAsUI：如果为 **True**，将显示“另存为” 对话框。

Cancel：如果为 **False**，则事件发生。如果事件过程将该参数设为 **True**，则过程完成后文档不会保存。

4. DocumentChange 事件

该事件在创建新文档、打开已有文档或激活其他文档时发生。

语法：

Private Sub object_DocumentChange()

object： 在类模块的事件中声明的 Application 类型事件。

5. DocumentOpen 事件

该事件在文档打开时发生。

语法：

Private Sub object_DocumentOpen(ByVal *Doc* As Document)

object：在类模块的事件中声明的 Application 类型对象。

Doc ：将打开的文档。

6. NewDocument 事件

该事件在创建新文档时发生。

语法：

Private Sub object_NewDocument(ByVal *Doc* As Document)

object：在类模块的事件中声明的 Application 类型对象。

Doc：新文档。

7. Quit 事件

该事件在用户退出 Word 时发生。

语法：

Private Sub **object**_Quit()

object：　在类模块的事件中声明的 Application 类型对象。

8. WindowActivate 事件

文档窗口激活时，该事件发生。

语法：

Private Sub object_WindowActivate(ByVal Doc As Word.Document, ByVal Wn As Word.Window)

object：　在类模块的事件中声明的 Application 类型对象。

Doc：仅用于 Application 对象。在活动窗口中显示的文档。

Wn：　将被激活的窗口

本示例在文档窗口被激活时，将其最大化显示。

```
Private Sub App_WindowActivate _
 (ByVal Wn As Word.Window)
 Wn.WindowState = wdWindowStateMaximize
End Sub
```

9. WindowBeforeDoubleClick 事件

如果在默认的双击操作发生前双击文档窗口的编辑区，该事件发生。

语法：

Private Sub object_WindowBeforeDoubleClick(ByVal Sel As

Selection, ByVal Cancel As Boolean)

object：在类模块的事件中声明的 Application 类型对象。

Sel：当前选定内容。

Cancel：如果为 False，则事件发生。如果事件过程将该参数设为 True，则过程完成时，默认双击操作不会发生。

10. WindowBeforeRightClick 事件

该事件在默认的右键单击操作发生之前，右键单击文档窗口的编辑区时发生。

语法：

Private Sub object_WindowBeforeRightClick(ByVal Sel As Selection, ByVal Cancel As Boolean)

object：在类模块的事件中声明的 Application 类型对象。

Sel：当前选定内容。

Cancel：如果为 False，则事件发生。如果事件过程将该参数设为 True，则过程完成时默认右键单击操作不会发生

11. WindowDeactivate 事件

该事件在文档窗口成为非活动窗口时发生。

语法

Private Sub **object**_WindowDeactivate(ByVal **Doc** As Word.Document, ByVal **Wn** As Word.Window)

object：在类模块的事件中声明的 Application 类型对象。

Doc：仅用于 Application 对象。非活动窗口中显示的文档。

Wn：非活动窗口。

12．WindowSelectionChange 事件

该事件在活动窗口的选定内容改变时发生。

语法：

Private Sub object_WindowSelectionChange(ByVal Sel As Selection)

object：在类模块的事件中声明的 Application 类型对象。

Sel：新的选定内容。

8.2 documents 对象和 document 对象集合

documents 对象集合和 document 对象都是 Application 对象的子对象。document 对象代表的是一篇完整的 Word 文档。它包括了文档中所有的对象，如段落、文本、字、句、表格、格式等。这些对象之间也是有鲜明的层次的，而 documents 对象集合则是所有的 document 对象的集合。

我们可以通过“对象浏览器”或者 Word2000 的在线帮助来查看 documents 对象集合和 document 对象的属性和方法。

我们可以通过如下几种方法得到 Document 对象：

通过使用 Documents 对象集合以及名称或索引序号来得到文档对

象，其语法为 Documents(*index*)。其中 index 既可以是文档的名称（如“mydoc.doc”），也可以是文档的索引序号。

例如，用 Documents("Example.doc")可以返回当前文档集合中名为“Example.doc”的文档，

索引序号代表文档在 Documents 集合中的位置。下列示例激活 Documents 集合中的第一篇文档。

```
Documents(1).Activate
```

使用属性和方法来返回 Document 对象，例如，使用 Activedocument 属性可以得到当前活动的文档该对象引用活动文档（具有焦点的文档）。用 Docuemnts 对象集合的 Add 和 Open 方法可以分别新建和打开一个文档，并返回这个文档对象。

如下例打开一篇文档，并将文档窗口的最小化。

```
Set MyDoc=Documents.Open（C:\my documents\example.doc）
Mydoc.Windows.State=wdWindowStateMinilize
```

下面，我们将讲述如何通过使用 Documents 对象集合和 Document 对象的方法和事件来操作 Word 中的文档。

8.2.1 Document 对象和 Documents 对象集合的方法

下面将分类讲述使用 Document 对象和 Documents 对象集合的方法对 Word 文档进行的操作。

1. 新建文档

使用 Documents 对象集合的 **Add** 方法可以新建一篇空白文档，并将返回的 document 对象添至 Documents 对象集合之中。

Add 方法的语法如下：

expression.Add(Template, NewTemplate, DocumentType, Visible)

expression：代表一个 Documents 对象集合，不可省略。

Template：Variant 类型，指定新文档使用的模板名。如果忽略此参数，则使用 Normal 模板。

NewTemplate：Variant 类型，如果此属性设置为 True，则将文档作为模板打开。默认值为 False。

DocumentType：Variant 类型，其值可取下列 WdNewDocumentType 常量之一：

WdNewBlankDocument　　创建一个空白文档,默认值

WdNewEmailMessage　　新建一个电子邮件信息

wdNewWebPage 新建一个 Web 页

Visible：Variant 类型，如果此参数为 True，Microsoft Word 将在可见窗口打开文档。如果此参数为 False，Word 仍会打开此文档，但文档窗口的 **Visible** 属性变为 False。默认值为 True。

下面的示例新建一个以 Normal.dot 为模板的空白文档。

```
Documents.Add
```

上面的示例只是简单地创建一个文档，如果我们想要对创建的文档作进一步的操作的话，我们最好将 Add 方法返回的 document 对象赋予一个对象变量。然后通过操作这个对象变量，我们就很容易对创建的新文档进行控制。

如下例所示，我们创建一个新文档并且将文档的段落格式设置为居中对齐，缺省字体设为“宋体”，并将文档保存为“文档 1.doc”。

```
Set newDoc = Documents.Add
With newDoc
.Content.Font.Name = "Arial"
.ActiveDocument.Content.ParagraphFormat.Alignment = wdAlignParagraphCent
.SaveAs FileName:="文档 1.doc"
End With
```

2. 打开文档

要打开一篇现有的文档，可使用 Documents 集合的 Open 方法。

使用Open方法可以打开指定的文档并将其添至 Documents 集合。整个表达式返回一个 Document 对象。

Documents.Open(FileName,ConfirmConversions,ReadOnly,AddToRecentFiles,

PasswordDocument,PasswordTemplate,Revert,WritePasswordDocument,

WritePasswordTemplate, Format, Encoding, Visible)

上式中各项的意义分别如下：

FileName：Variant 类型，代表文档名（可包含路径），不可省略。

ConfirmConversions：Variant 类型，该属性为 True 时，如果文档不是 Microsoft Word 格式，则显示“文件转换”对话框。

ReadOnly：Variant 类型，如果此属性为 True，则以只读方式打开文档。

AddToRecentFiles：Variant 类型，可选。如果本属性为 True，则会将上述文件名添至“文件”菜单底部的最近使用过的文件列表中。

PasswordDocument：Variant 类型，打开此文档时所需的密码。

PasswordTemplate：Variant 类型，打开此模板时所需的密码。

Revert：Variant 类型，用于控制当 *FileName* 与已打开的文档同名时执行的操作。如果此属性为 True，则放弃对已打开文档进行的所有尚未保存的改动，并将重新打开该文档。如果此属性为 False，则激活已打开的文档。

WritePasswordDocument：Variant 类型，保存对文档进行的更改时所需的密码。

WritePasswordTemplate：Variant 类型，保存对模板进行的更改时所需的密码。

Format：Variant 类型，打开文档时使用的文件转换器。可为下列

WdOpenFormat 常量之一：wdOpenFormatAllWord、wdOpenFormatAuto、wdOpenFormatDocument、wdOpenFormatEncodedText、wdOpenFormatRTF、wdOpenFormatTemplate、wdOpenFormatText、wdOpenFormatUnicodeText 或 wdOpenFormatWebPages。默认值为 wdOpenFormatAuto。

Encoding：Variant 类型，在查看保存文档时，Microsoft Word 所使用的文档编码（代码页或字符集）。可以是任何有效的 MsoEncoding 常量。可以在“Visual Basic 编辑器”的“对象浏览器”中查看有效 MsoEncoding 常量的列表。默认值为系统代码页。Long 类型，可读写。

Visible：Variant 类型，可选。如果此值为 True，则在可见窗口中打开文档。默认值为 True。

本示例以只读方式打开文档 MyDoc.doc。

Documents.Open FileName:="C:\MyFiles\MyDoc.doc", ReadOnly:=True

下面的指令打开名为 **MyDocument.doc** 的文档（该文档位于“MyFolder”文件夹中）。

Documents.Open FileName:="C:\MyFolder\MyDocument.doc"

3. 保存文档

要保存一篇文档，可使用 Document 对象的 Save 方法。

使用 Save 方法，我们既可以保存某一个指定的文档，也可以同时保存 Documens 对象集合中的所有文档，还可以将文档另存为别的名字。其语法分别如下：

（1）保存某个指定文档

保存某一指定文档的语法为：

expression.save

整个表达式返回一个 document 对象（文档）或者一个 template 对象（模板）。表达式中各项的含义分别为：

expression:　可以为 document 对象或 template 对象，代表要保存的文档或者模板

下面的指令保存名为 Sales.doc 的文档。

Documents("Sales.doc").Save

✍ 注意：Documents("Sales.doc")是一个 document 对象

（2）保存所有打开的文档

通过使用 Documents 对象集合的 Save 方法，我们可以同时保存所有打开的文档。其语法格式如下：

Documents.Save(NoPrompt, OriginalFormat)

NoPrompt：Variant 类型，如果为 True，Word 将自动保存所有文

档。如果为 False，当一篇文档在上次存盘后进行了修改时，Word 将提醒用户分别保存各文档。

OriginalFormat：Variant 类型，指定文档保存的方式。可为以下 WdOriginalFormat 常量之一：wdOriginalDocumentFormat、wdPromptUser 或 wdWordDocument。

下面的示例将按照文档原有的格式保存 Documents 集合中的每一文档，但在保存文档前不提示用户。

```
Documents.Save NoPrompt:=True, _
OriginalFormat:=wdOriginalDocumentFormat
```

（3）另存文档

要将文档另存为别的格式，可使用 Document 对象的 SaveAs 方法。其语法如下：

Document.SaveAs(FileName，FileFormat，Password, WritePassword,)

整个表达式返回一个 Document 对象。

FileName：Variant 类型，代表要将文档另存的文件名。它既可以只是文件名，也可包含完整的路径（例如，“C:\Documents\Temporary File.doc”）。其默认值为当前文件夹和文件名。如果从未保存过此文档，则将使用默认的文件名（例如，文档 1.doc）。如果已有 FileName 指定的文档，则覆盖此文档，而且在覆盖前不提醒用户。

FileFormat：Variant 类型，代表文档保存的格式。其值可以为以下 WdSaveFormat 常量之一：wdFormatDocument、wdFormatDOSText、wdFormatDOSTextLineBreaks 、 wdFormatEncodedText 、 wdFormatHTML、wdFormatRTF、wdFormatTemplate、wdFormatText、wdFormatTextLineBreaks 或 wdFormatUnicodeText。

Password：Variant 类型，打开文档时的口令。

WritePassword：Variant 类型，保存对文档的修改所需的口令。

下面的示例将当前的活动文档以缺省名保存在当前的文件夹。

ActiveDocument.SaveAs

4. 关闭文档

通过使用 Document 对象或者 Documents 对象集合的的 Close 方法，我们可以关闭某个指定的文档或者所有打开的文档。Close 方法的语法如下：

expression.Close(SaveChanges, OriginalFormat)

整个表达式返回一个 document 对象或者 documents 对象集合。

expression：可以是一个 Document 对象或 Documents 对象集合。

SaveChanges： Variant 类型，指定保存文档时的操作。可选下列 WdSaveOptions 常 量 之 一 ： wdDoNotSaveChanges 、 wdPromptToSaveChanges 或 wdSaveChanges。

OriginalFormat： Variant 类型，可选。指定文档的保存格式。可选

下列 WdOriginalFormat 常量之一：wdOriginalDocumentFormat、wdPromptUser 或 wdWordDocument。

下面的指令关闭并保存当前活动文档。

```
Activedocument.Close SaveChanges:=wdSaveChanges
```

下面的示例关闭所有打开的文档，并一一提示用户是否保存。

```
Documents.Close SaveChanges:= wdPromptToSaveChanges
```

5. 激活文档

要激活一个文档，可以使用 Document 对象的 Activate 方法。有关 Activate 方法的语法在前面讲述 Appliation 对象时已经介绍过了。

在下面的示例中，如果“文档 1.doc”文档是打开的，则下面的示例激活该文档，如果没有打开该文档，则显示一个对话框。

```
For Each MyDoc In Documents
 If InStr(1, MyDoc.Name, "sample.doc", 1) Then
  aDoc.Activate
 Else
  docFound = False
 End If
Next aDoc
If docFound = False Then _
Msgbox “Document not open”
```

8.2.2　使用 document 对象的事件

1. New 事件

New 事件在创建一个基于模板的文档时发生。只有当 New 事件的过程储存于该模板中时此事件才可运行。其语法如下：

Private Sub Document_New()

下面的示例的作用是：当创建基于模板的新文档时重排所有的打开文档。

```
Private Sub Document_New()
Application.Windows.Arrange wdTiled
End Sub
```

✍ 注意：必须将 New 事件的过程保存在模板之中

2. Open 事件

当打开一个文档时，Open 事件发生。其语法如下：

Private Sub Document_Open()

本示例在文档打开时显示 Word 程序的使用者名称。

```
Private Sub Document_Open()
Msgbox Application.UserName
End Sub
```

3. Close 事件

当关闭一篇文档时，Close 事件发生。其语法为：

Private Sub Document_Close()

本示例可实现的功能是：文档关闭时，在文件服务器上创建文档的备份副本。（此过程储存在文档中，而非模板中）。

```
Private Sub Document_Close()
ThisDocument.Save
ThisDocument.SaveAs "\\network\backup\" & ThisDocument.Name
End Sub
```

8.3 Range 对象

使用 Visual Basic 通常需要完成的任务是在文档中指定一个区域，然后对该区域进行一些操作，例如插入文字或应用格式。例如，可能需要编写一个宏，用来在文档的某一部分查找一个单词或词组。文档的这一部分就可以用 Range 对象来代表。定义 Range 对象后，就可以应用 Range 对象的方法和属性来修改这个区域的内容。

Range 对象代表文档中的一个范围。与在文档中使用的书签类似，在 Visual Basic 过程中使用 Range 对象可以定义文档的某一部分。每一个 Range 对象由一起始和一终止字符位置定义。一个 Range 对象小至只是一个插入点，大至包括整篇文档。但是与书签不同，Range 对象只在定义该对象的过程运行时才存在。

例如，在下面的示例中，我们在过程 Test1 中定义了一个 Range 对象 MyRange，我们可以在过程 Test1 中对这个 Range 对象进行操作；因为 Range 对象只在定义该对象的过程运行时才存在，所以在另外一个过程 Test2 中，我们不能对 MyRange 对象进行操作。

```
Sub test1()
    Set myrange = ActiveDocument.Range(Start:=1, End:=10)
    myrange.InsertBefore "hello"
    End Sub
    Sub test2()
    Myrange.InsertBefore“hello”                    '本行会出现错误
    End Sub
```

Range 对象和文档中的选定内容是相互独立的，跟文档的选定内容对应的是 Selection 对象。在每一窗体中只能有一个选定内容，但我们可以在文档中定义多个范围。对范围进行定义和复制不需要改变文档的选定内容。

8.3.1 引用 Range 对象

通过 Range 属性，Range 方法以及 SetRange 方法，我们可以得到 Range 对象。

1. 用 Range 属性得到 Range 对象

有多种对象具有 Range 属性，例如 Paragraph、Bookmark 和 Cell 对象等等，用 Range 属性可以返回一个 Range 对象，此 Range 对象返回调用 Range 方法的对象在文档中对应的那一部分。

例如，下例返回一个 Range 对象，该对象引用活动文档的第一个段落。

```
Set myRange = ActiveDocument.Paragraphs(1).Range
```

下例返回一个 Range 对象，该对象引用文档中的选定部分，并将选定部分的字体改为黑体，格式改为段落居中。

```
Sub GetRange()
Set myrange = Selection.Range
    myrange.Font.Bold = True
    myrange.ParagraphFormat.Alignment = wdAlignParagraphCenter
End Sub
```

2. 使用 Range 方法得到 Range 对象

Range 方法可以被 Document 对象调用，使用 Range 方法可以在指定的文档中创建并返回一个 Range 对象。该对象对应于已给定起点和终点的主要文字部分。

Range 方法的语法如下：

object.Range(Start, End)

其中 object 代表一个 Document 对象。

Start：Long 类型，开始字符的位置，其缺省值为 0

End：Long 类型，结束字符的位置，其缺省值为文档末尾的位置。

其中，Start 的值如果为 m，则表示 Range 对象的开始位置在文档的第 m 个字符之后；End 的值如果为 n，则表示 Range 对象的结束位置在文档的第 n 个字符之后。如在下面的语句中，MyRange 对象的范围为从文档的第一百个字符到文档的第一千个字符。

```
Set myRange = ActiveDocument.Range(Start:=100, End:=1000)
```

在上面我们已经讲过，Range 对象大可以到整篇文档，小到可以是一个插入点。在下面的示例中，我们创建了两个 Range 对象，第一个对象是一个插入点，我们利用它在文档中插入一些文字，第二个 Range 对象引用整篇文档，我们用它修改全文的格式：

```
Sub ()
        Set myrange1 = activedocuement.Range(0, 0)
        myrange1.InsertBefore "Written by Michael"
        Set myrange2 = activedocuement.Range
        myrange2.Font.Italic = True
    End Sub
```

我们还可以使用诸如 Selection、Bookmark 或 Range 之类的对象

的 Start 和 End 属性来定义 Range 对象的开始和结束位置。

在下面的示例中，创建 Range 对象，该对象从第二段的开头开始，至第三段的末尾结束。

```
Set myRange =ActiveDocument.Range(Start:=myDoc.Paragraphs(2).Range.Start, _
    End:=myDoc.Paragraphs(3).Range.End)
```

使用 Start 和 End 属性跟使用字符编号其实是一样的，不信请运行下面的例子：

```
Sub atest1()
        MsgBox ActiveDocument.Paragraphs(1).Range.End
    End Sub
```

我们得到如下图所示的对话框，可见 Start 和 End 属性返回的仍然是段落的首尾位置在全文档中的字符编号。

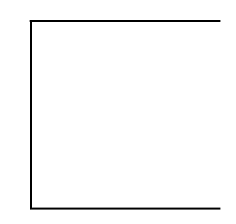


图 8-4　显示第一段末尾的字符编号

3. 重新定义一个 Range 对象

- SetRange 方法用以重新定义原有 Selection 或 Range 对象的起始字符和结束字符位置。

✍ 注意：字符位置的值从文档该部分开头计起，起始值为 0。将计算所有的字符，包括非打印字符和未显示的隐藏字符。

SetRange 方法的语法如下：

Object.SetRange(Start, End)

Object：必需。该表达式代表一个 Range 或 Selection 对象。

Start：必需。Long 类型，指区域或所选内容的起始字符位置。

End：必需。Long 类型，指区域或所选内容的结束字符位置。

下例使用 SetRange 方法重新 定义一个现有的 Range 对象。下面的示例将 myRange 定义为当前所选内容。SetRange 方法重新定义 myRange，使该变量可引用当前所选内容及其后的 10 个字符。

```
Set myRange = Selection.Range
myRange.SetRange Start:=myRange.Start, End:=myRange.End + 10
```

✍ SetRange 方法不同于 Range 方法，Range 方法将创建一个 Range 对象，并给出 Range 对象的起始和结尾位置;SetRange 方法不创建 Range 对象，只是对原有的 Range 对象重新设置开始和结束位置。

本示例用 SetRange 方法重新定义 myRange 的范围，使其代表从文档的第二句到选定内容的结尾位置之间的文档。

```
Sub RangeSet()
```

```
Set myRange = ActiveDocument.Paragraphs(1).Range
myRange.SetRange
Start:=Activedocument.Sentenses(2).Range.Start, _
End:=ActiveDocument.Selection.End
End Sub
```

由于 Range 对象对应的是某一特定位置的文档，我们可以对 Range 对象应用可对文档使用的任何属性和方法。通过对 Range 对象的操纵，我们可以改变 Range 对象对应的文档的属性。

8.3.2 Range 对象的使用

Range 对象引用的是文档中的某一个连续区域，通过对 Range 对象的操作，我们可以改变引用区域的属性。

下面是几个对 Range 对象进行操作的示例。

示例 1：将 Range 对象剪贴到文档的末尾

```
Sub CutRange()
    Set Myrange=ActiveDocument.Selection.Range
    Myrange.Cut
    Myrange.SetRange
    Start=ActiveDocument.Paragraphs(2).Range.End,_
```

```
    End=ActiveDocument.Paragraphs(2).Range.End
    Myrange.Paste
End Sub
```

示例 2：本示例复制已指定给 myRange 变量的 Range 对象。然后将所复制的区域折叠到结尾点，扩展一个字符单位，并将该字符设为大写。再为源 Range 对象（即 myRange）设置倾斜格式。

```
Sub Example()
    Set myRange = Selection.Range
With myRange.Duplicate
    .Collapse Direction:=wdCollapseEnd
    .Expand Unit:=wdCharacter
.Case = wdUpperCase
End With
myRange.Font.Italic = True
End Sub
```

8.4　Selection 对象

用 Word 对文档进行处理时，经常会先选定文字，然后对选定部分进行操作（如设置文字的格式或键入文字）。如果需要通过代码与选定内容相对应或要更改所选内容时，就应使用 Selection 对象进行操

作

Selection 对象代表窗体中的选定内容。每个窗体中只能有一个 Selection 对象，而且只能激活一个 Selection 对象。这跟 Range 对象是不同的。

跟 Range 对象相同，Selection 对象代表的选定内容既可以是文档中的一个区域，也可以仅仅是一个插入点。

8.4.1 如何引用 Selection 对象

要引用 Selection 对象，我们可以使用 Select 方法选定指定的对象，而后用 Selection 属性返回该 Selection 对象。

1. 通过 Select 方法选定指定的对象

用 Select 方法，我们可以选定指定的对象，其语法如下：

Object.Select

其中 Object 代表要选定的对象。

例如，我们可以用下面的语句来选定整个文档

```
Activedocument.Select
```

下面的语句可以选定文章的第一段

```
Activedocument.Paragraphs(1).Range.Select
```

下例选定 Table1.doc 中第二张表格的第一列。

```
Documents("Table.doc").Tables(2).Columns(1).Select
```

2. 用 Selection 属性返回 Selection 对象

Selection 属性是一个只读属性，它返回一个 Selection 对象。

本示例将第一个窗口的所选内容复制到下一个窗口。

```
If Windows.Count >= 2 Then
 Windows(1).Selection.Copy
 Windows(1).Next.Activate
 Selection.Paste
End If
```

在下面的示例中，如果插入点不在表格中，则将所选内容（插入点）移至下一张表格。

```
If Selection.Information(wdWithInTable) = False Then
 Selection.GoToNext What:=wdGoToTable
End If
```

下面的示例将底纹应用于选定内容中每张表格的首行。

```
If Selection.Tables.Count >= 1 Then
 For Each aTable In Selection.Tables
 aTable.Rows(1).Shading.Texture = wdTexture10Percent
 Next aTable
End If
```

8.4.2　使用 Selection 对象

Selection 对象引用的是当前文档中的选定部分。如果改变了 Selecition 对象的属性，文档中选定部分的属性也会随之改变。

Selection 对象的重要属性如下：

1. Type 属性

用 Selection 对象的 **Type** 属性可以操作选定内容类型（例如，选定内容是一块内容还是一个插入点）。

下例当选定内容为插入点时将选定内容变为当前段，并将选定内容格式设置为阴文

```
If Selection.Type = wdSelectionIP Then
  Selection.Paragraphs(1).Range.Select
  Selection.Font.Engrave = True
End If
```

2. Information 属性

用 Information 属性可以返回选定内容的有关信息。下例当选定内容在表格中时显示该表格的行数和列数。

```
If Selection.Information(wdWithInTable) = True Then
  MsgBox "Columns = " _
  & Selection.Information(wdMaximumNumberOfColumns) _
  & Visual BasicCr & "Rows = " _
```

```
& Selection.Information(wdMaximumNumberOfRows)
End If
```

3. Range 属性

用 Range 属性可从 Selection 对象返回一个 Range 对象。

下面的将变量 myRange 定义为所选范围，并将边框应用于选定内容的所有段落

```
Set myRange = Selection.Range
MyRange.Paragraphs.Borders.Enable = True
```

8.5 Dialog 对象

Dialog 对象代表一个内置对话框。它 Dialogs 集合的一个成员。Dialogs 集合包含了 Word 中所有的内置对话框。通过使用 Dialog 对象，我们可以调用 Word 2000 的内置对话框，实现用户和程序之间的交互。

✍ Dialogs 集合是所有 Word 2000 内置对话框的集合。我们不能新建一个内置对话框或将其加入 Dialogs 集合。

8.5.1 引用 Dialog 对象

用 Dialogs（index）可以返回一个 Dialog 对象，此处 index 为一个标志对话框的 WdWordDialog 常量。

下例显示和执行来自“文件”菜单“打开”内置对话框的操作。

dlgAnswer=Dialogs(wdDialogFileOpen).Show

✍ WdWordDialog 常量由“wdDialog”作前缀，后面跟菜单和对话框的名称而建立的。例如，“页面设置”对话框的该常量为 wdDialogFilePageSetup，而“新建”对话框的该常量为 wdDialogFileNew。

8.5.2 Dialog 对象的属性和方法

1. Show 方法

使用 Dialog 对象的 Show 方法可以显示显示并执行任何内置 Word 对话框的操作。要访问特定的内置 Word 对话框，可将 Dialogs 属性赋值为 WdWordDialog 常量。

语法

expression.Show(TimeOut)

expression： 必需。代表一个 Dialog 对象。

TimeOut： Variant 类型，可选。此为 Word 在自动关闭对话框之前所等待的时间。单位为毫秒。如果省略这一参数，那么只有通过用户才能关闭对话框。

Show 方法的返回值为 Long 类型，用于指明关闭对话框时单击的按钮。

表 8-1　Show 方法返回值的描述

返回值	描述
-2	“关闭”按钮。
-1	“确定”按钮。
0（零）	“取消”按钮。
> 0（零）	命令按钮，1 代表第一个按钮，2 代表第二个按钮，以此类推。

例如，下面的宏指令显示“打开”对话框(wdDialogFileOpen)。

```
Dialogs(wdDialogFileOpen).Show
```

选定文件并单击“确定”按钮后，文件被打开（执行操作）。

本示例显示“查找和替换”对话框，并在该对话框的“查找内容”文本框中预置单词“Blue”。如果用户在 5 秒钟内不执行 “查找” 操作的话，对话框将自动关闭。

```
With　Dialogs(wdDialogEditFind)
.Find = "Blue"
.Show　Timeout:=5000
End With
```

设置 DefaultTab 属性可以访问 Word 对话框的特定制表位。下面的示例显示“边框和底纹”对话框（位于“格式”菜单）中的“页边框”制表位。

```
WithDialogs(wdDialogFormatBordersAndShading)
.DefaultTab=wdDialogFormatBordersAndShadingTabPageBorder
```

.Show

EndWith

2. Display 方法

Display 方法的语法如下：

expression.Display(TimeOut)

expression： 必需。该表达式将返回一个 Dialog 对象。

TimeOut： Variant 类型，可选。此项为自动关闭对话框前 Word 的等待时间。每一单位大约为 0.001 秒。如果省略这一参数，对话框只有在用户关闭它时才被关闭。系统的并发运行可能会增加有效时间值。

Display 方法看上去跟 Show 方法很类似，它用于显示对话框，但是它不能执行对话框的操作。如果要用内置对话框来提示用户并返回设置，该方法很有用。

下例将显示“选项”对话框中的“用户信息”选项卡。并且返回和显示用户名称。

WithDialogs(wdDialogToolsOptionsUserInfo)

.Display

MsgBox.Name

EndWith

执行上面的语句，Word 2000 会显示“用户信息”对话框。如果我

们在显示的对话框中将“用户姓名”修改为“YOURS”，在关闭对话框后，我们发现对话框的设置没有改变（如图 8-1 所示），这就是 Display 方法和 Show 方法的区别所在。

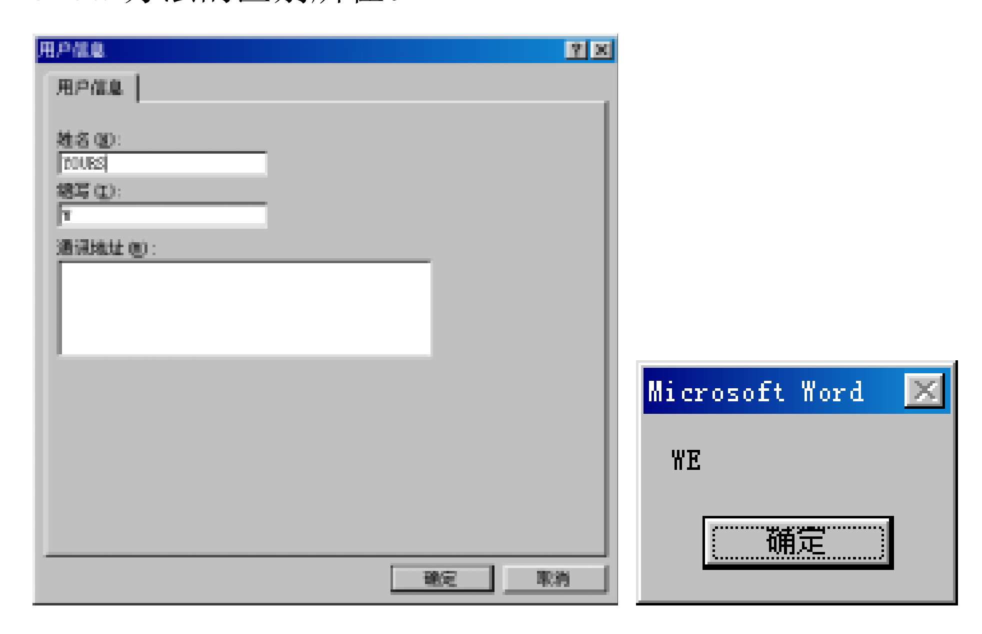


图 8-1　用“Display”方法显示的对话框

3. Execute 方法

Execute 方法跟 Display 方法恰恰相反，它可以执行某个对话框中的设置而不显示该对话框。

Execute 方法的语法如下：

expression.Execute

expression：必需。代表一个 Dialog 对象

下面的示例用于选中“段落”对话框的“换行和分页”选项卡中的“与下段同页”复选框，注意：在整个过程中对话框是不显示的。

```
With Dialogs(wdDialogFormatParagraph)
.KeepWithNext = 1
.Execute
End With
```

4. DefaultTab 属性

在显示具有多个选项卡的对话框时，可以用 DefaultTab 属性返回或设置活动的选项卡。其中 DefaultTab 属性的值可以是 WdWordDialogTab 常量中的任何一个。

✍ 要浏览 WdWordDialogTab 常数的列表，可以打开“Visual Basic 编辑器”的“对象浏览器”进行查看。

下面的示例将显示所选的“页面设置”对话框中的“纸张来源”选项卡，如图 8-2 所示。

```
With Dialogs(wdDialogFilePageSetup)
.DefaultTab = wdDialogFilePageSetupTabPaperSource
.Show
End With
```

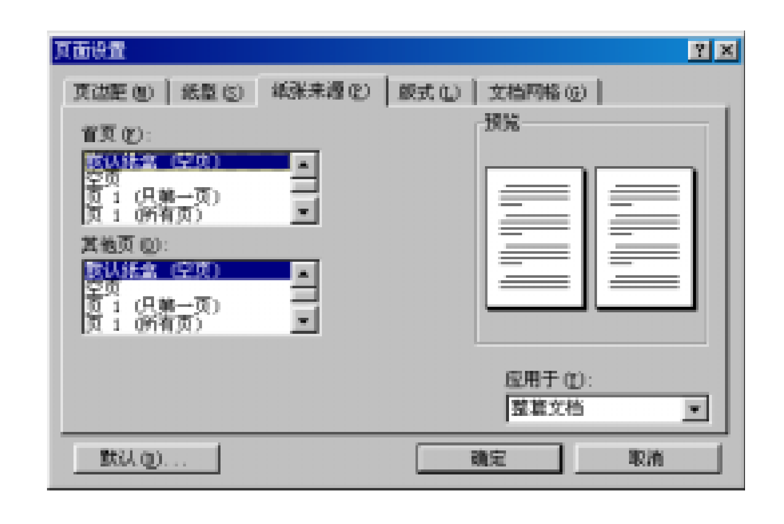


图 8-2　显示“纸张来源”选项卡

8.6　Tasks 对象集合和 Task 对象

Tasks 对象集合是由 Task 对象组成的集合，它代表系统当前运行的所有任务。

可用 Application 对象的 Tasks 属性返回一个 Tasks 集合。

可使用 Shell 函数运行可执行程序并将其添入 Tasks 集合。其语法如下：

Shell(pathname[,windowstyle])

Shell 函数的语法含有下面这些命名参数：

Pathname：　必要参数。代表要执行的程序名，它可以包括任何必需的参数或命令行变量，还可以包括目录或文件夹，以及驱动器名。

Windowstyle：可选参数。表示在程序运行时窗口的样式。如果

windowstyle 省略，则程序是以具有焦点的最小化窗口来执行的。

下例使用 Shell 函数来运行一个用户指定的应用程序。' 将第二个参数值设成 1，可让该程序以正常大小的窗口完成，并且拥有焦点。

```
Dim RetVal
RetVal = Shell("C:\WINDOWS\CALC.EXE", 1)    ' 完成 Calculator
```

8.6.3 Tasks 对象集合的属性和方法

Tasks 对象集合的属性和方法有：

1．Exists 方法

判断指定书签或任务是否存在。如果存在指定的书签或任务，则返回 True。

Exists 方法的语法如下：

expression.Exists(Name)

上式中各参数的含义如下：

expression：该表达式返回一个 Bookmarks 或 Tasks 对象。

Name：String 类型，必需。代表任务的名称。

下例用于判断 Windows 计算器程序（如果存在该任务）是否正在运行。如果计算器没有运行，Shell 语句将启动此程序。如果计算器正在运行，将激活此应用程序。

```
If Tasks.Exists("Calculator") = False Then
.Shell "Calc.exe"
Else
Tasks("Calculator").Activate
End If
Tasks("Calculator").WindowState = wdWindowStateNormal
```

2. ExitWindows 方法

ExitWindows 方法用于关闭所有打开的应用程序，退出 Microsoft Windows，注销当前用户。该方法不保存对打开的 Word 文档所做的修改，但提示您保存在其他基于 Windows 的应用程序中对打开的文档所做的修改。

该方法的语法如下：

expression.ExitWindows

expression：必需。该表达式将返回一个 Tasks 对象。

8.6.4 引用 Task 对象

用 Tasks（index）返回一个 Task 对象，其中 index 为应用程序名或索引号.

下例打开 Tasks 集合中第一个可视任务的应用程序窗口并改变其尺寸。

```
With Tasks(1)
If .Visible = True Then
.Activate
.Width = 400
.Height = 200
End If
End With
```

下例的功能是：如果“计算器”程序在 Tasks 集合中，则恢复该应用程序窗口。

```
If Tasks.Exists("Calculator") = True Then
Tasks("Calculator").WindowState = wdWindowStateNormal
End If
```

8.6.5　Task 对象的属性和方法

1．Left、Top、Width、Height 属性

Left 属性和 Top 属性用于设置任务窗口的位置；Width 属性和 Height 属性用于设置任务窗口的大小，它们都以磅为单位。

本示例将活动窗口的水平位置设置为 100 磅，宽度设为 200 磅。

```
With ActiveDocument.ActiveWindow
.Width=200
```

.Left = 100

.Top = 0

End With

2. Visible 属性

Visible 属性用于设置某项任务是否可见。

在下例中，如果计算器正在运行，则本示例将其隐藏；如果计算器没在运行，则显示一则消息。

If Tasks.Exists("Calculator") Then

Tasks("Calculator").Visible = False

Else

Msgbox "Calculator is not running."

End If

3. WindowState 属性

WindowState 属性用于返回或设置指定的文档窗口或任务窗口的状态。可选下列 WdWindowState 常量之一：wdWindowStateMaximize、wdWindowStateMinimize 或 wdWindowStateNormal。

本示例最小化 Microsoft Excel 应用窗口。

For Each myTask In Tasks

If InStr(myTask.Name, "Microsoft Excel") > 0 Then

myTask.Activate

```
myTask.WindowState = wdWindowStateMinimize
End If
Next myTask
```

- 不能设置非活动窗口的状态。在设置窗口状态之前可用 Activate 方法激活窗口。

4. Activate 方法

Activate 方法用于激活某个任务，其语法如下：

expression.Activate

上式中各参数的意义如下：

expression：必需。代表一个 Task 对象。

Wait：Boolean 类型，可选。如果本参数为 True，则只在用户激活 Word 之后才会激活任务。如果本参数为 False，则即使 Word 不处于活动状态，也可以立即激活该任务。

如果“记事本”应用程序在 Tasks 集合中，则本示例激活“记事本”。

```
For Each myTask In Tasks
If InStr(myTask.Name, "Notepad") > 0 Then
myTask.Activate
myTask.WindowState = wdWindowStateNormal
End If
Next myTask
```

5. Close 方法

Close 方法用于关闭指定的任务，其语法如下：

expression.Close

在上式中，各参数的意义如下：

expression：必需，代表要关闭的任务。

本示例先激活 Microsoft Excel，然后再将其关闭。

```
For Each myTask In Tasks
If InStr(myTask.Name, "Microsoft Excel") > 0 Then
myTask.Activate
myTask.Close
End If
Next myTask
```

8.7 System 对象

System 对象用于返回有关计算机系统的信息。

通过使用 Application 对象的 System 属性，我们可以得到 System 对象。

8.7.1 System 对象的属性和方法

System 对象的主要属性和方法有：

1. FreeDiskSpace 属性

用 FreeDiskSpace 属性可以返回当前驱动器的磁盘可用空间，以字节为单位。

✍ 用 ChDir 语句可改变当前驱动器。

下例检查磁盘可用空间的大小。如果可用空间小于 10MB，则显示一条提示信息。

```
If (System.FreeDiskSpace \ 1048576) < 10 Then _
MsgBox "Low disk space"
```

2. HorizentalResolution 属性和 VerticalResolution 属性

HorizentalResolution 属性和 VerticalResolution 属性用于返回显示器的水平分辨率和垂直分辨率，单位为象素。

本示例显示当前的屏幕分辨率（例如 1024 x 768）。

```
horz = System.HorizontalResolution
vert = System.VerticalResolution
MsgBox "Resolution = " & horz & " x " & vert
```

3. OperatingSystem 属性

OperatingSystem 属性返回当前操作系统的名称，如“Windows”或“Windows NT”。String 类型.

本示例显示包括有当前操作系统名称的信息。

```
MsgBox "This computer is running " & System.OperatingSystem
```

4. Connect 方法

用 Connect 方法可以建立到网络驱动器的连接。其语法如下：

expression.Connect(Path, Drive, Password)

在上式中，各参数的意义如下：

expression：必需。该表达式返回一个 System 对象。

Path：String 类型，必需。网络驱动器的路径（例如，“\\hst\games”）。

Drive：Variant 类型，可选。此数对应于指定给网络驱动器的字母，0 对应第一个有效驱动器字母，1 对应第二个有效驱动器字母，以此类推。如果省略此参数，则使用下一个有效字母。

Password：Variant 类型，可选。如果网络驱动器有密码保护，则指定密码。

同时使用 Dialogs 属性和 wdDialogConnect 常量可显示“映射网络驱动器”对话框。下面示例显示“映射网络驱动器”对话框，预设路径。

```
With Dialogs(wdDialogConnect)
.Path = "\\Marketing\Public"
.Show
End With
```

下例建立到以口令“smiley”保护的网络驱动器（\\Project\Info）的连接，并将下一个有效驱动器字母指定给网络驱动器。

```
System.Connect Path:="\\Project\Info", Password:="smiley"
```

第九章　定制 Word 的菜单和工具栏

首先，我们要了解在 Office2000 中，与菜单和工具栏有关的对象。

9.1　菜单和工具栏对应的对象

9.1.1　CommandBar 对象和 CommandBars 对象集合

CommandBar 对象代表的是应用程序中的一个命令栏。所谓命令栏，指的是在 VBA 中用于控制菜单和工具栏的可编程对象，例如菜单栏，工具栏，子菜单，快捷菜单都是 ComandBar 对象。

用 CommandBars(index) 可返回一个 CommandBar 对象；此处 index 是该命令栏的名字或索引号。

以下示例逐个筛选命令栏集合中的每个成员，以查找名为“Forms”的命令栏（“窗体”工具栏）。如果找到，那么本示例显示该命令栏并保护其固定状态。在本示例中，变量 cb 代表一个 CommandBar 对象。

```
Sub Example()
    foundFlag = False
    For Each cb In CommandBars
```

```
If cb.Name = "Forms" Then
cb.Protection = msoBarNoChangeDock
cb.Visible = True
foundFlag = True
End If
Next cb
If Not foundFlag Then
MsgBox "The collection does not contain a Forms command bar."
End If
End Sub
```

在上面我们已经讲过，ComandBar 对象可以是一个工具栏，菜单栏，也可以是一个菜单，快捷菜单或者子菜单。我们可用名字或索引号指定菜单栏或工具栏（这些菜单栏或工具栏位于容器应用程序的有效菜单栏和工具栏列表中），但是对于菜单，子菜单和快捷菜单而言，我们只能用它的名称来指定。也就是说，我们不可能用“CommandBars（数字）”这样的形式来得到一个菜单。

在下面示例可实现：在“工具”菜单底部添加一个新的菜单项。单击该菜单项将运行 h 宏“Macro1”。

```
Sub Example()
Set newItem =
```

```
CommandBars("Tools").Controls.Add(Type:=msoControlButton)
    With newItem
     .BeginGroup = True
     .Caption = "Make Report"
     .FaceID = 0
     .OnAction = "macro1"
    End With
End Sub
```

如果两个或两个以上的自定义菜单或子菜单具有相同的名字，那么 CommandBars(index) 返回第一个具有该名字的菜单。为确保返回正确的菜单或子菜单，可找到显示该菜单的弹出式控件，然后对该弹出式控件用 CommandBar 属性以返回代表该菜单的命令栏。

> ✍ 弹出式控件相当于一个“父菜单”或者一个“父控件”，它有两种类型：一种是位于菜单栏或者工具栏上的内制控件或用户自定义控件，单击会弹出一个菜单；另一种是位于菜单，子菜单，快捷菜单上的内制菜单或者用户自定义菜单，用鼠标放在上面时会显示其子菜单。

在下例中，假定工具栏“Custom Tools”中第三个控件是一个弹出式控件，本示例可实现将 **Save** 命令添加在该菜单的底部。

```
Set viewMenu = CommandBars("Custom Tools").Controls(3)
```

```
viewMenu.Controls.Add ID:=3     'ID of Save command is 3
```

CommandBar 对象的常用属性和方法有：

1. Context 属性

Context 是一个只读属性，String 类型，它返回或设置一个字符串，该字符串确定指定命令栏的保存位置。在 Word2000 中，它返回附加在包含指定 CommandBar 对象的文档上的模板的完整名称（包括磁盘路径）。在所有其他 Microsoft Office 程序中，该属性将返回一个空字符串，或者程序不支持该属性。

✍ 只能对自定义命令栏设置 **Context** 属性。

在下面的示例中将显示一个消息框，该消息框显示了命令栏“Custom”的保存位置。

```
Sub Example()
    Set myBar = CommandBars _
      .Add(Name:="Custom", Position:=msoBarTop, _
      Temporary:=True)
    With myBar
      .Controls.Add Type:=msoControlButton, ID:=2
      .Visible = True
    End With
    MsgBox (myBar.Context)
```

End Sub

运行上面的过程，得到如图 9-1 所示的对话框。



图 9-1　显示工具栏的保存位置

2. Enabled 属性

Enabled 属性是一个只读属性，Boolean 类型，它用于控制工具栏或者控件的工作状态。如果 Enabled 属性值为 True 的话，则指定的命令栏或命令栏控件处于激活状态，如果其值为 False 的话，则命令栏或者命令栏控件处于无效状态。此时我们不能选择该命令栏（命令栏控件）。

对于命令栏，如果将该属性设置成 True，那么该命令栏的名字将出现在有效命令栏列表中。

对于内置控件，如果将 Enabled 属性设置成 True，那么将由应用程序确定其状态（我们不可能在没有选定文档内容的情况下执行 Copy 命令）。但如果将该属性设置成 False，那么该控件无效。

下例可实现的功能为：统计当前文档中表格的数目，如果表格的数目为零，则将“表格”菜单栏设为无效；若文档中表格的数目不为零，则将激活“表格”菜单栏。

```
Sub Example()
    If ActiveDocument.Tables.Count >= 1 Then
      CommandBars("table").Enabled = True
      Else: CommandBars("table").Enabled = False
End Sub
```

3. Name 属性和 NameLocal 属性

Name 属性是一个只读属性，String 类型，它返回或设置指定的 **CommandBar** 对象的名字。

NameLocal 是一个可读写属性，String 类型，它用于返回内置的命令栏的名字，该名字用应用程序的版本语言显示。用 NameLocal 属性也可返回或设置自定义命令栏的名字。

✍ 对内置命令栏，**Name** 属性返回该命令栏的美式英语名字。用 **NameLocal** 属性可返回其本地化的名称。

如果改变了一个自定义命令栏的 **LocalName** 属性值，那么该命令栏的 **Name** 属性值也将改变，反之亦然

在下例中，将分别显示“Edit”菜单的 Name 属性和 NameLocal 属性。

```
Sub Example()
    With CommandBars(“Edit”)
      MsgBox "The name of the command bar is " & .Name
```

```
        MsgBox "The localized name of the command bar is " & .NameLocal
    End With
End Sub
```

显示的结果如图 9-2 所示。



图 9-2　内置工具栏“Edit”的 Name 属性



图 9-3　内置工具栏“Edit”的 NameLocal 属性

4．Visible 属性

Visible 属性是一个 Boolean 类型的可读写属性，主要用于设置 CommandBar 和 CommandBarControl 对象的可见性。如果 Visible 属性的值为 True，那么指定的命令栏或命令栏控件将显示于屏幕上；如果属性的值为 False，即使命令栏或者命令栏控件处于激活状态，我们在屏幕上也看不见它们。

新建自定义命令栏的 Visible 属性的默认值为 False。

✍ 只有先将命令栏的 Enabled 属性设置为 True，才可将其 Visible 属性设置为 True。

5. Delete 方法

我们可以用 Delete 方法删除一个 CommandBar 对象，其语法如下：

expression.Delete

expression： 不可省略，整个表达式返回一个 CommandBar 对象。

下例的功能是：删除所有非内置且不可见的命令栏。

```
Sub Example()
   foundFlag = False
   delBars = 0
   For Each bar In CommandBars
    If (bar.BuiltIn = False) And _
    (bar.Visible = False) Then
    bar.Delete
    foundFlag = True
    delBars = delBars + 1
    End If
   Next bar
   If Not foundFlag Then
    MsgBox "No command bars have been deleted."
```

```
    Else
      MsgBox delBars & " custom bar(s) deleted."
    End If
End Sub
```

6. FindControl 方法

FindControl 方法用于在集合中查找符合条件的 **CommandBarControl** 对象（我们后面将会讲到）。其语法如下：

expression.FindControl(Type, Id, Tag, Visible, Recursive)

各项的意义如下：

expression：不可省略。表达式返回一个 CommandBars 对象。

Type：Variant 类型，可选。指代要查找的控件类型。如 ActiveX 控件（msoControlActiveX），按钮控件（msoControlButton）等等。想要详细地得到 Type 的可选值，可以查询 Microsoft Office 2000 的帮助。

Id：Variant 类型，可选。指代要查找的控件的标识符。

Tag：Variant 类型，可选。指代要查找的控件的标签值。

Visible：Variant 类型，可选。其默认值为 False。如果该值为 True，那么只查找屏幕上显示的命令栏控件（所有可见的工具栏和菜单栏）。

Recursive：Boolean 类型，可选。如果该值为 True，那么将在命令栏及其全部弹出式子工具栏中查找。默认值为 False。

如果 CommandBars 集合中有两个或者更多的控件符合搜索条

件，那么 FindControl 返回找到的第一个控件。如果没有控件符合搜索条件，那么 FindControl 返回 Nothing。

本示例可实现的功能为：在“常用”工具栏中查找第一个按钮控件，并新建一个“FirstButton”工具栏，将这个按钮控件放在新建的工具栏中

```
Sub Example()
 Set newCtrl = CommandBars(“Standard”).FindControl(Type:= _
 MsoControlButton)
 CommandBars.add(name:=”FirstCtrl”,Position:=msoBarPopup)
 CommandBars(“FirstCtrl”).ShowPopup
End Sub
```

顾名思义，CommandBars 对象集合是 CommandBar 对象的集合。用 CommandBars 属性可返回一个 CommandBars 集合。

CommandBars 对象集合的主要属性、方法和事件有：

7. Add *方法*

用 Add 方法可在 CommandBars 集合中添加一个新的命令栏，并返回一个 Commandbar 对象。

Add 方法的语法如下：

```
expression.Add(Name, Position, MenuBar, Temporary)
```

各项的意义分别为：

expression：不可省略，整个表达式返回一个 **CommandBars** 对象。

Name：Variant 类型，代表新命令栏的名字。如果省略该参数，那么 Word 给该命令栏指定一个默认值。（例如“Custom 1”）。

Position：Variant 类型，代表新命令栏的位置。该参数值可设置为以下 MsoBarPosition 常量之一：

表 9-1 参数 Position 的可选值

常量	描述
MsoBarLeft，msoBarTop，msoBarRight，msoBarBottom	指示新命令栏的左右和上下坐标。
MsoBarFloating	表明新命令栏是可移动的。
MsoBarPopup	表明新命令栏是一个快捷菜单。
MsoBarMenuBar	Macintosh 专有

MenuBar： Variant 类型，可选。为 True 时，用新命令栏替换活动命令栏。默认只为 False。

Temporary：Variant 类型，可选。如果此参数值为 True，那么新命令栏只是暂时有效，当应用程序关闭时，将删除此命令栏。该参数的默认值为 False。Microsoft Word 和 Microsoft Outlook 会忽略此参数，因此在退出应用程序之前，您必须人工删除所有临时添加到 CommandBars 集合中的项目。

本示例可实现的功能为：添加一个位于最高端的名为“Custom”的

命令栏。本示例还可在命令栏中添加一个内置按钮，此按钮的功能为“粘贴”。

```
Set mybar = CommandBars _
 .Add(Name:="Custom", Position:=msoBarTop, _
 Temporary:=True)
With mybar
 .Controls.Add Type:=msoControlButton, Id:= _
 CommandBars("Edit").Controls("Paste").ID
 .Visible = True
End With
```

8．ActiveMenuBar 属性

使用 CommandBars 集合的 ActiveMenuBar 属性可以返回一个 **CommandBar** 对象，该对象指代引用程序中的活动菜单栏。ActiveMenuBar 也是一个只读属性。

在下例中：活动菜单栏的末尾暂时添加一个菜单“Custom”，然后在该菜单添加一个菜单项“command1”。

```
Sub Example()
   Set myMenuBar = CommandBars.ActiveMenuBar
   Set newMenu = myMenuBar.Controls.Add(Type:=msoControlPopup,
```

```
        Temporary:=True)
        newMenu.Caption = "Custom"
        Set ctrl1 = newMenu.CommandBar.Controls _
          .Add(Type:=msoControlButton, Id:=1)
        With ctrl1
          .Caption = "command1"
          .TooltipText = "command1"
          .Style = msoButtonCaption
        End With
    End Sub
```

9. FindControl 方法和 FindControls 方法

CommandBars 集合除了具有 FindControl 方法之外，还有 FindControls 方法。FindControls 方法用于查找符合条件的 CommandBarControls 集合。

FindControls 方法的语法跟 FindControl 方法的语法是相同的，这里不再赘述。

10. OnUpdate 事件

Word2000 的命令栏（包括菜单栏和工具栏）发生任何改变时，都会触发 OnUpdate 事件。

OnUpdate 事件的语法如下：

Private Sub CommandBars_OnUpdate()

- OnUpdate 事件由 CommandBar 对象和所有命令栏控件识别。对命令栏或命令栏控件的任何改变或者命令栏或命令栏控件状态的任何改变都会触发该事件。所以在使用此事件时应多加注意。

9.1.2 CommandBarControl 对象

CommandBarControl 对象代表的是一个命令栏控件。所谓命令栏控件，指的是位于工具栏，菜单栏，菜单，子菜单和快捷菜单上的控件（包括内置控件和用户自定义控件）。

由图 9-3 可知：CommandBarControl 对象可以分为三种：CommandBarButton 对象（按钮控件），CommandBarComboBox 对象（复合框构件）和 CommandBarPopup 对象（弹出式构件）。CommandBarControl 对象具有的所有属性和方法，CommandBarButton、CommandBarComboBox 以及 CommandBarPopup 对象都具有；而 CommandBarButton 对象（或者是 CommandBarComboBox、 CommandBarPopup 对象）具有的属性，CommandBarControl 对象不一定具有。

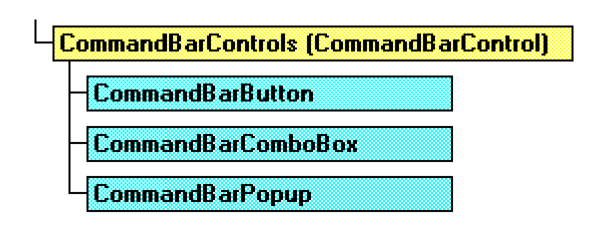


图 9-4　CommandBarControl 对象的结构模型

- 要为自定义命令栏控件编写 Visual Basic 代码，可使用 CommandBarButton 、 CommandBarComboBox 和 CommandBarPopup 对象。要为容器应用程序中的内置控件写代码，而该控件又不能用上述三个对象中的任意一个表示，那么可使用 CommandBarControl 对象。

要引用 CommandBarControl 对象，我们可以使用下面两种方法：

1．用 Controls(index) 可返回一个 CommandBarControl 对象；此处 index 是该控件的索引号。（该控件的 Type 属性必须是 msoControlLabel 、 msoControlExpandingGrid 、 msoControlSplitExpandingGrid 、 msoControlGrid 或 msoControlGauge。）

2．用 CommandBars 或者 CommmandBar 对象的 FindControl 方法也可返回一个 CommandBarControl 对象。

我们知道，CommandBarButton 代表的是一个按钮控件，CommandBarComboBox 代表的是一个复选框控件，

CommandBarPopup 代表的是一个弹出式控件。下面，我们将讲述通用控件对象常用的属性和方法，而 CommandBarControl 对象代表的是一个通用的控件。下面，我们先来讲述通用的控件属性和方法。对于按钮控件，复选框控件和弹出式控件的特有的属性和方法，我们将放在后面讲述。

1. BuiltIn 属性

BuiltIn 是一个只读的 Boolean 类型的属性，它用于判定一个命令栏（命令栏控件）是程序内置的还是用户自定义的。属性值如果为 **True**，那么指定命令栏或命令栏控件是应用程序的内置命令栏或控件。如果属性值为 **False**，那么它可能是自定义命令栏或控件，也可能是已设置了 OnAction 属性的内置控件。

本示例删除所有未显示的自定义命令栏。

```
Sub DelCustomcontrol()
  foundcontrol = False
  deletedcontrols = 0
  For Each mycontrol In CommandBars(1).controls
    If (mycontrol.BuiltIn = False)    Then
      control.Delete
      foundcontrol = True
      deletedControls = deletedControls + 1
```

```
    End If
  Next
  If Not foundcontrol Then
    MsgBox "No controls have been deleted in the Standard Bar"
  Else
    MsgBox deletedControls & " custom control(s) deleted."
  End If
End Sub
```

2. Caption 属性

Caption 是一个 String 类型的属性，代表指定控件的标题。Caption 属性是一个可读写的属性，因此我们可以用它来设置控件的标题。

本示例可实现的功能为：在自定义命令栏中添加一个带拼写检查按钮图符的命令栏控件，然后将其标题设置为“Spelling checker”。

```
Sub SetCaption()
  Set myBar = CommandBars.Add(Name:="Custom", _
  Position:=msoBarTop, Temporary:=True)
  myBar.Visible = True
  Set myControl = myBar.Controls _
  .Add(Type:=msoControlButton, Id:=2)
  With myControl
```

```
    .DescriptionText = "Starts the spelling checker"
    .Caption = "Spelling checker"
  End With
End Sub
```

3. OnAction 属性

OnAction 属性是一个 String 类型的可读写属性，它主要用于定义单击控件时发生的动作。它的返回值是一个过程，如果不存在此过程，程序会出现错误。

如果我们给某个控件设置了 OnAcion 属性，那么在用户单击或更改此控件的值时，将会运行 OnAction 属性返回的过程。

如下例：在命令栏“Custom”中添加一个命令栏控件，该控件被单击时将运行名叫“MySub”的过程。

```
Sub AddControl()
    Set myBar = CommandBars("Custom")
    Set myControl = myBar.Controls _
    .Add(Type:=msocontrolButton)
    With myControl
    .FaceId = 2
    .OnAction = "MySub"
    End With
```

```
    myBar.Visible = True
End Sub
```

4. Priority 属性

Priority 属性是一个可读写的 Long 类型的属性。通过使用控件的 Priority 属性，可以返回或设置指定控件的优先级。工具栏中的每个控件都有一个优先级，对于固定的命令栏，当一行中无法容纳其中的所有控件时，优先级较低的控件将会从命令栏中略去。

有效的优先级数从 0（零）到 7，其他优先级的值将被忽略。优先级小于等于 1 则表示该控件不能从工具栏中删除。

✍ 只有工具栏中的控件才有 Priority 属性，菜单项的栏控件不能使用 Priority 属性。

本示例移动一控件并将其优先级设为 5，这样如果在命令栏的一行中无法容纳所有控件，那么该控件将有可能被略去。

```
Sub SetControlPriority()
    Set allcontrols = CommandBars("Format").Controls
    For Each ctrl In allControls
      If ctrl.Type = msoControlComboBox Then
      With ctrl
      .Move Before:=7
      .Tag = "Selection box"
```

```
        .Priority = 5
        End With
        Exit For
      End If
      Next
End Sub
```

5. Copy 方法

使用 Copy 方法可以将一个控件复制到某个命令栏中。

Copy 方法的语法如下：

```
expression.Copy(Bar, Before)
```

各项的意义分别如下：

expression：代表要拷贝的控件，不可省略。整个表达式返回一个 CommandBarControl、CommandBarButton、CommandBarPopup 或 CommandBarComboBox 对象。

Bar：Variant 类型，代表要将控件拷贝到的目标命令栏。如果省略该参数，控件将复制到同一个命令栏中（即该控件原来所在的命令栏）。

Before：Variant 类型，代表新控件在目标命令栏上位置。新控件将插在原先居于该位置的控件之前。如果省略此参数，那么控件将被复制到命令栏的末尾。

在下例中：将命令栏“Standard”中第一个控件复制到命令栏

“Custom”中，并仍将其放置在第一个位置。

```
Sub CopyControl()
    Set myCustomBar = CommandBars("Custom")
    Set myControl = CommandBars("Standard").Controls(1)
    With myControl
      .Copy Bar:=myCustomBar, Before:=1
      .SetFocus
    End With
End Sub
```

6. Delete 方法

使用 CommandBarControl 对象的 Delete 方法，可以在包含该控件对象的集合中删除该对象。

```
expression.Delete(Temporary)
```

各项的意义分别如下：

expression： 不可省略，整个表达式一个 CommandBarControl 对象

Temporary： Variant 类型。如果参数值为 True，那么在当前时段删除该控件。应用程序将在下一个时段重新显示该控件。如果参数值为 False，则永久删除该控件。

7. Execute 方法

使用控件的 Execute 方法，我们可以运行该控件对应的命令。如果控件是应用程序的内置控件，使用 Execute 方法执行控件的内置命令；如果控件是用户自定义的，则运行控件的 OnAction 属性对应的过程。

Execute 方法的语法如下：

expression.Execute

整个方法返回一个 CommandBarControl、CommandBarButton、CommandBarPopup 或 CommandBarComboBox 对象。

在下例中，如果当前文档在上次修改之后没有保存过，则保存当前文档。

```
Sub SaveDoc()
    If ActiveDocument.Saved = False Then
    Set mybar = CommandBars(1).Controls(3)
    mybar.Execute
    End If
End Sub
```

8. Move 方法

使用 Move 方法，可以将控件移动到某个工具栏中。

Move 方法的语法如下

语法：

expression.Move(Bar, Before)

上式中各项的意义如下：

expression： 代表要拷贝的控件，不可省略。整个表达式返回一个 CommandBarControl 、 CommandBarButton 、 CommandBarPopup 或 CommandBarComboBox 对象。

Bar：Variant 类型，代表要将控件拷贝到的目标命令栏。如果省略该参数，控件将复制到同一个命令栏中（即该控件原来所在的命令栏）。

Before：Variant 类型，代表新控件在目标命令栏上位置。新控件将插在原先居于该位置的控件之前。如果省略此参数，那么控件将被复制到命令栏的末尾。

- Move 方法和 Copy 方法的区别在于: 使用 Move 方法移动控件之后，控件将不会在原先的工具栏中出现

9. SetFocus 方法

使用 SetFocus 方法，可以将键盘的焦点移到某一个指定控件。当然，如果该控件在当前状态下是无效的或者是不可见的，那么 SetFocus 方法无效。

当某个控件获得焦点时，该控件将会被三维突出显示。此时可以按箭头键将在工具栏的控件之间转移焦点。按下回车键，可以执行该控件对应的命令和方法，并取消控件的焦点；按下 Esc 键，可以直接取消控件的焦点。

SetFocus 方法的语法如下：

```
expression.SetFocus
```

SetFocus 方法将返回调用该方法的控件对象。

下面是一个简单的宏，它将键盘焦点转移到“常用”工具栏的“新建”工具按钮上。

```
Sub Simple()
    CommandBars(1).Controls(3).SetFocus
End Sub
```

运行完该宏之后，注意观察一下工具栏上的“新建”工具按钮。此时如果我们按下回车键，就可以新建一篇文档。

按钮控件特有的属性，方法和事件有：

（1）CopyFace 和 PasteFace 方法

CopyFace 方法用于将指定的按钮控件的图符复制到“剪贴板”上，PasteFace 方法用于将“剪贴板”的内容复制到指定的按钮控件上。其语法格式分别为：

```
expression.CopyFace
expression.PasteFace
```

用 CopyFace 方法和 PasteFace 方法将会返回一个 CommandBarButton 对象。

在下例中，我们在 Custom 工具栏中新建一个按钮控件，并对其应

用“保存文档”按钮的图标。

```
Sub GetSaveIcon()
    Set mybar = CommandBars _
     .Add(Name:="Custom", Position:=msoBarTop, _
     Temporary:=True)
     mybar.Visible = True
     Set mybutton = mybar.Controls.Add(Type:=msoControlButton)
     CommandBars(1).Controls(3).CopyFace
    mybutton.PasteFace
End Sub
```

（2）Click 事件

当用户单击一个按钮控件时，Click 事件发生。

Click 事件的语法如下：

```
Private Sub CommandBarButton_Click(ByVal Ctrl As CommandBarButton,
ByVal CancelDefault As Boolean)
```

上式中两个参数的意义如下：。

Ctrl：CommandBarButton 类型，不可省略。此参数指代初始化 Click 事件的按钮控件（即单击的按钮控件）。

CancelDefault：Boolean 类型，不可省略。其值为 False 时执行与 CommandBarButton 控件关联的默认操作。

Click 事件由 CommandBarButton 对象识别。要返回特定 CommandBarButton 控件的 Click 事件，请用关键词 WithEvents 声明一个变量，并将变量的值设置为该控件。

在下例中，在宿主应用程序的“文件”菜单中创建了一个新命令栏按钮，它使用户能以 CSV (逗号分隔）文件格式保存工作簿。（本示例可在所有应用程序中运行，但有关存为 CSV 格式的内容只适用于 Microsoft Excel。）

```
Private HostApp As Object
Sub createAndSynch()
 Dim iIndex As Integer
 Dim iCount As Integer
 Dim fBtnExists As Boolean
 Dim obCmdBtn As Object

 On Error GoTo errHandler
  Set HostApp = Application
  Dim barHelp As Office.CommandBar
 Set barHelp = Application.CommandBars("File")
```

```
fBtnExists = False
iCount = barHelp.Controls.Count
For iIndex = 1 To iCount
   If barHelp.Controls(iIndex).Caption = "Save As CSV (Comma Delimited)" Then fBtnExists = True
 Next
Dim btnSaveAsCSV As Office.CommandBarButton
If fBtnExists Then
   Set btnSaveAsCSV = barHelp.Controls("Save As CSV (Comma Delimited)")
Else
   Set btnSaveAsCSV = barHelp.Controls.Add(msoControlButton)
   btnSaveAsCSV.Caption = "Save As CSV (Comma Delimited)"
End If
 btnSaveAsCSV.Tag = "btn1"
btnSaveAsCSVHandler.SyncButton btnSaveAsCSV
Exit Sub

errHandler:
```

```
  ' Insert error handling code here
End Sub
```

上述示例依靠以下代码，此代码保存在 VBA 工程的类模块中。

```
Private WithEvents ButtonClickEvent As Office.CommandBarButton
Private Sub ButtonClickEvent_Click(ByVal Ctrl As Office.CommandBarButton, _
  CancelDefault As Boolean)
  ActiveWorkbook.SaveAs "MyWorkbook.csv", xlCSV
  CancelDefault = True
End Sub

Public Sub SyncButton(btn As Office.CommandBarButton)
  Set ButtonClickEvent = btn
  If Not btn Is Nothing Then
    MsgBox "Synced '" & btn.Caption & "' button events."
  End If
End Sub
Private Sub Class_Terminate()
  Set ButtonClickEvent = Nothing
```

End Sub

复选框控件特有的属性，方法和事件有：

（1）AddItem 方法和 RemoveItem 方法

使用 AddItem 方法和 RemoveItem 方法可以在用户自定义的组合框控件中添加或者删除一个列表项。

使用 AddItem 方法和 RemoveItem 方法将返回一个组合框对象。

- 应当注意的是，AddItem 方法和 RemoveItem 方法只能用于组合框或者下拉框，不能用于编辑框或内置的组合框控件。

AddItem 方法和 RemoveItem 方法的语法如下：

expression.AddItem(Text, Index)

expression.RemoveItem(Index)

上式中各项的意义如下：

expression：不可省略。指代要执行操作的组合框控件。

Text：String 类型，不可省略。指代要添加到指定控件中的项。

Index:：Variant 类型，可选。指定要添加或者删除的项在列表各项中的位置。如果省略此参数，指定项将添加在列表的末尾

在下例中，在命令栏中添加两个个组合框控件，然后在第一个列表中添加两个选项。第二个列表中先添加两个选项，再删除一个。

Sub AddDeleteItem()

```
Set myBar = CommandBars("Custom")
Set myControl1 = myBar.Controls.Add(Type:=msoControlComboBox, Id:=1)
With myControl1
  .AddItem "First Item", 1
  .AddItem "Second Item", 2
End With
Set myControl2 = myBar.Controls.Add(Type:=msoControlComboBox, Id:=1)
With myControl2
  .AddItem "First Item", 1
  .AddItem "Second Item", 2
    .DeleteItem 2
End With
End Sub
```

（2）Change 事件

当最终用户改变组合框控件中的选项时，Change 事件发生。

Change 事件的语法如下：

Private Sub CommandBarComboBox_Change(ByVal Ctrl As CommandBarComboBox)

上式中各项的意义如下：

Ctrl：不可省略，代表引发改事件的组合框控件。

Change 事件由 CommandBarComboBox 对象识别。要返回某个 CommandBarComboBox 控件的 Change 事件，请用关键词 WithEvents 声明一个变量，并将其值设置为该 CommandBarComboBox 控件。当触发 Change 事件时，会执行该控件的 OnAction 属性所指定的宏或代码

以下示例创建一个包含 CommandBarComboBox 控件的命令栏，该控件包含四个选项。该组合框通过 CommandBarComboBox_Change 事件响应用户操作。

```
Private ctlComboBoxHandler As New ComboBoxHandler
Sub AddComboBox()
 Set HostApp = Application
 Dim newBar As Office.CommandBar
 Set newBar = HostApp.CommandBars.Add(Name:="Test CommandBar", Temporary:=True)
 Dim newCombo As Office.CommandBarComboBox
 Set newCombo = newBar.Controls.Add(msoControlComboBox)
 With newCombo
 .AddItem "First Class", 1
```

```
  .AddItem "Business Class", 2
  .AddItem "Coach Class", 3
  .AddItem "Standby", 4
  .DropDownLines = 5
  .DropDownWidth = 75
  .ListHeaderCount = 0
  End With
  ctlComboBoxHandler.SyncBox newCombo
  newBar.Visible = True
End Sub
```

上述示例依靠以下代码，此代码保存在 VBA 工程的类模块中。

```
Private WithEvents ComboBoxEvent As Office.CommandBarComboBox
Public Sub SyncBox(box As Office.CommandBarComboBox)
  Set ComboBoxEvent = box
  If Not box Is Nothing Then
    MsgBox "Synced " & box.Caption & " ComboBox events."
  End If
End Sub
Private Sub Class_Terminate()
```

```
  Set ComboBoxEvent = Nothing
End Sub
Private Sub ComboBoxEvent_Change(ByVal Ctrl As
Office.CommandBarComboBox)
  Dim stComboText As String
    stComboText = Ctrl.Text
        Select Case stComboText
      Case "First Class"
          FirstClass
      Case "Business Class"
          BusinessClass
      Case "Coach Class"
          CoachClass
      Case "Standby"
          Standby
  End Select

End Sub
Private Sub FirstClass()
  MsgBox "You selected First Class reservations"
```

```
End Sub
Private Sub BusinessClass()
  MsgBox "You selected Business Class reservations"
End Sub
Private Sub CoachClass()
  MsgBox "You selected Coach Class reservations"
End Sub
Private Sub Standby()
  MsgBox "You chose to fly standby"
End Sub
```

顾名思义，CommandBarControls 集合是 CommandBarControl 对象的集合。CommandBarControl 对象代表的是一个控件，CommandBarControl 对象集合代表一个命令栏（菜单栏，工具栏）中所有的控件的集合。

使用 CommandBar 对象或者 CommandBarPopup 对象的 Controls 属性可返回一个 CommandBarControls 集合。

在下例中，我们将“Standard”工具栏中所有控件的设为无效。

```
Sub DisableAll()
    Set AllControls = CommandBars(“Standard”).Controls
    For Each SingleControl in AllControls
```

```
        SingleControl.Enabled=False
    Next
End Sub
```

运行上面的宏，你会发现……。不要紧，将上面的过程的第四行中的 False 改为 True，再一次运行该宏，就可以把 Word 恢复到原来的样子了。

使用 CommandBarControls 集合的 Add 方法可在集合中添加一个新的控件。如下面的语句可在命令栏“Custom”中新添一个空白按钮。

```
Set myBlankBtn = CommandBars("Custom").Controls.Add
```

9.2 定制 Word 的工具栏和菜单栏

在上面的章节中，我们已经简述了有关 Word 的命令栏的一些基本的对象，属性，方法和事件。有了这些知识，我们就可以在 Word2000 中定制自己的命令栏了。

9.2.1 添加/删除自定义工具栏

要在 Word 中添加一个自定义工具栏，可以使用 CommandBars 集合的 Add 方法。其操作如下：

（1）打开“工具”菜单中“宏”子菜单，选择“宏”命令。

（2）在对话框中输入要创建的宏的名字，这里是

“AddCommandBar”。

（3）单击创建按钮，进入 Visual Basic 编辑器。

（4）在宏中输入代码如下：

```
Sub AddCommandBar()
    Set mybar = CommandBars _
     .Add(Name:="Custom", Position:=msoBarfloating, _
     Temporary:=True)
End Sub
```

上例在 Word 中暂时加入一条名为“Custom”的浮动工具栏。在本次 Word 的运行过程中，“Custom”工具栏是存在的。但是在关闭 Word 时，Word 会自动将此工具栏删除。要想永久创建一条工具栏，可以将 Add 方法中的 Temporary 参数改为 False，如下：

```
Set mybar = CommandBars _
    .Add(Name:="Custom", Position:=msoBarfloating, _
Temporary:=False)
```

如果想要修改变工具栏的名称和位置，可以对应地修改 Add 方法中的 Name 和 Position 参数。具体内容请参照本章前面的内容。

如果读者运行了上面的程序，可能会发现找不到我们创建的“Custom”工具栏，这是因为新创建的工具栏的 Visible 属性都被设置为 False。要想看到“Custom”工具栏，我们只需在上面的过程的最后

再加上一条语句即可：

```
mybar.visible=true
```

通过下面的代码，我们可以删除刚刚创建的“Custom”工具栏。

```
Sub DeleteBar()
    Set Mybar = CommandBars(“Custom”)
    Mybar.Delete
End Sub
```

工具栏一经删除，就不能再恢复。

9.2.2　向工具栏中添加控件

上面我们讲述了如何在 Word 中创建一个工具栏，我们还亲手创建了一个“Custom”工具栏。我们创建的仅仅是一个空的工具栏，如果在一条工具栏中没有控件，那么这条工具栏就没有任何意义。下面我们将学习如何向工具栏中添加控件。

下面的代码将在“Custom”工具栏中加入一个按钮控件。

```
expression.Add(Type, Id, Parameter, Before, Temporary)
    Sub AddControl()
        Set mybar = CommandBars（”Custom”）
         mybar.Visible = True
         Set                      mybutton                      =
```

```
mybar.Controls.Add(Type:=msoControlButton)
End Sub
```

用上面的代码，可以向“Custom”工具栏中暂时添加一个按钮控件。要想永久添加一个控件，可将程序修改如下：

```
Sub AddControl()
    Set mybar = CommandBars（”Custom”）
     mybar.Visible = True
     Set mybutton =
mybar.Controls.Add(Type:=msoControlButton _
    Temporary:=False)
End Sub
```

通过修改 Add 方法中 Type 参数的值，我们可以在工具栏中加入五种不同类型的控件。其对应关系如表 9-2 所示。

表 9-2　Type 参数的可选值

Type 参数的值	**对应的控件类型**
MsoControlButton	按钮控件
MsoControlEdit	编辑框控件
MsoControlDropDown	下拉列表框控件
MsoControlComboBox	组合框控件
MsoControlPopup	弹出式控件

在上面的代码中创建的按钮控件是一个“空白”的按钮，我们还应该给按钮加上图标，以便于区分各个按钮。

要给按钮加上一个“图标”，我们可以使用按钮控件的 FaceId 属性，见下面的代码：

```
Sub AddControl()
    Set mybar = CommandBars（”Custom”）
     mybar.Visible = True
     Set mybutton =
mybar.Controls.Add(Type:=msoControlButton _
    Temporary:=False)
    Mybutton.FaceId=CommandBars(“Standard”).Controls(3).Fac
eId
End Sub
```

在上面的代码中，我们使用“常用”中“保存文件”按钮的图标作为新建按钮“mybutton”的图标。

此外，我们还可以使用按钮控件的 CopyFace 和 PasteFace 方法来更改按钮的图标。

```
Sub AddControl()
    Set mybar = CommandBars（”Custom”）
     mybar.Visible = True
     Set mybutton =
mybar.Controls.Add(Type:=msoControlButton _
```

```
        Temporary:=False)
        CommandBars(“Standard”).CopyFace
        Mybutton.PasteFace
    End Sub
```

上面的代码同样将“常用”中“保存文件”按钮的图标作为新建按钮“mybutton”的图标。其效果跟使用 FaceId 属性是完全相同的。

我们已经向工具栏中添加了一个按钮，还给这个按钮设置了图标，但是这个按钮到现在仍然是“中看不中用”。我们还必需给按钮添加一定的功能，使得单击这个按钮时，Word 能执行一定的操作。

对于新建的按钮而言，我们可以使它具有跟某个 Word 内置控件相同的功能，也可以给它自定义一个功能。

通过使用按钮控件的 Id 属性，可以让新建的按钮具有和 Word 中某个内置控件相同的功能。

ID 属性决定了控件的内置动作，所有用户自定义控件的 ID 属性值都是 1。

在下面的代码中，我们让新建的按钮控件具有和“常用”工具栏中“打印”按钮相同的功能。

```
    Sub AddControl()
        Set mybar = CommandBars（”Custom”）
```

```
  mybar.Visible = True
Set mybutton = mybar.Controls.Add(Type:=msoControlButton
_
Id:=CommandBars("Standard").Controls(5).ID,Temporary:=False)
CommandBars("Standard").CopyFace
Mybutton.PasteFace
End Sub
```

- 值得注意的是：ID 属性是一个只读属性，我们只能在在新建一个按钮的同时设置按钮的 ID 属性。类似下面的语句是错误的：

  Mybutton.Id=CommandBar("Standard").Controls(1).Id

如果想要自定义新建按钮的功能，我们可以使用控件的 OnAction 属性。在下面的例子中，我们新建了一个按钮，并且给按钮设置了如下的功能：当我们单击这个按钮时，将在程序的底部打开剪贴板工具栏：

```
Sub AddControl()
Set mybar = CommandBars（"Custom"）
  mybar.Visible = True
Set mybutton =
```

```
mybar.Controls.Add(Type:=msoControlButton)
    CommandBars(“Standard”).CopyFace
    Mybutton.PasteFace
    Mybutton.OnAction=”showclipboard”
End Sub

Sub ShowClipboard()
    CommandBars(“/clipboard”).position=msobarBottum
    CommandBars(“Clipboard”).visible=true
End Sub
```

在 Word 中不能建立一个永久性的自定义控件，因为在退出 Word 应用程序时所有的自定义控件都将被删除（并非所有的 Office 程序都是如此）。如果我们想要在退出 Word 程序之前手动删除某个控件的话，可以使用 CommandBarControl 对象的 Delete 方法。

在下例中，我们删除了刚才定义的“Custom”工具栏中的所有自定义控件。

```
Sub DeleteControl()
    For Each mycontrol in CommandBars(“Custom”).Controls
    If mycontrol.Id=1          Then
    Mycontrol.Delete
```

```
    Next
End Sub
```

9.2.3 创建和删除菜单

在缺省的情况下，我们使用 CommandBars 集合的 Add 方法创建的是一条工具栏。如何在 Word 中创建自定义的菜单呢。

要创建自定义的菜单，首先要得到一个菜单栏对象，通过 CommandBars 集合的 ActiveMenuBar 属性，我们可以得到 Word 中的菜单栏。菜单和子菜单都可以被看作是菜单栏中的控件，我们只需使用上面讲过的添加控件的方法即可在菜单栏中添加菜单。

在下例中，我们向 Word 的菜单栏中添加了一个“Select”菜单，在“Select”菜单中有两个菜单项“Select all”和“Select pragraphs”，在“Select pragraphs”菜单项中又有两个子菜单命令“Select paragraph 1”和“Select paragraph 2”。为了方便讲解，我们给语句加上了标号

```
Sub AddMenu()
1   Set myMenuBar = CommandBars.ActiveMenuBar
2   Set newMenu = myMenuBar.Controls.Add(Type:=msoControlPopup, Temporary:=True)
3   newMenu.Caption = "Select"
```

```
4   Set newmenuitem1 = newMenu.CommandBar.Controls _
        .Add(Type:=msoControlButton, ID:=1)
5   With newmenuitem1
6       .Caption = "select all"
7       .TooltipText = "select all"
8       .OnAction = "selectall"
9       .Style = msoButtonCaption
10  End With

11  Set newmenuitem2 = newMenu.CommandBar.Controls _
        .Add(Type:=msoControlPopup, ID:=1)
13  With newmenuitem2
14      .Caption = "select paragraph"
15      .TooltipText = "select paragraph"
16  End With

17  Set newsubmenu1 = newmenuitem2.CommandBar.Controls _
18      .Add(Type:=msoControlButton, ID:=1)
19  With newsubmenu1
20      .Caption = "paragraph 1"
```

```
21        .TooltipText = "select paragraph 1"
22        .Style = msoButtonCaption
23        .OnAction = "selectp1"
24  End With

25  Set newsubmenu2 = newmenuitem2.CommandBar.Controls _
26        .Add(Type:=msoControlButton, ID:=1)
27  With newsubmenu2
28        .Caption = "paragraph 2"
29        .TooltipText = "select paragraph 2"
30        .Style = msoButtonCaption
31        .OnAction = "selectp2"
32  End With

End Sub

Sub selectall()
    ActiveDocument.Select
End Sub
```

```
Sub selectp1()
    Set myrange = ActiveDocument.Paragraphs(1).Range
myrange.Select
End Sub

Sub selectp2()
    Set myrange = ActiveDocument.Paragraphs(2).Range
    myrange.Select
End Sub
```

各语句的功能如下：

第 1 句：　　得到一个菜单栏对象

第 2-3 句：　　创建 Select 菜单

第 4-10 句：　创建菜单命令“Select All”

第 11-16 句:　　创建菜单项“Select Paragraph”

第 17-24 句：创建子菜单命令“Select paragraph 1”

第 25-32 句：创建子菜单命令“Select paragraph 2”

- 值得注意的是：在添加菜单项时，如果我们想要让菜单项是一个命令，则应将它设为按钮控件-“msoControlButton”类型（见语句 4）；如果想要让这个菜单项的下一层还有子菜单项，则应将它设为弹出式控件-“msoControlPopup”

类型（见语句 11）。

同样，在 Word 退出时，会自动删除所有自定义的菜单。如果我们想要手动删除菜单，可以使用 CommandBarControl 对象的 Delete 方法。

下例可以删除所有自定义的菜单

```
Sub DeleteMyMenu()
    For Each mycontrol In CommandBars.ActiveMenuBar.Controls
        If mycontrol.ID = 1 Then
            mycontrol.Delete
        EndIf
    Next
End Sub
```

9.2.4 执行控件对应的操作

要在 VBA 中执行某个对应的操作，可以使用对应控件的 Execute 方法。

在下例中，执行了“视图”菜单中的“Web”版式命令

```
Sub Example()
    Set mymenu = CommandBars.ActiveMenuBar.Controls(3)
    mymenu.CommandBar.Controls(2).Execute
```

```
End Sub
```

第十章　用 VBA 操作 Word 文档

从前面的讲述中我们知道，Word 文档对应于一个 Document 对象。文档中的所有元素，在 VBA 中都有对象与之相对应。通过操纵这些对象，可以实现在 VBA 程序中对文档的控制。

常见的文档元素及对应的 VBA 对象如表 10-1 所示。

表 10-1　文档元素与相应的对象

文档元素	表达式	返回的对象
文档中单词集合	Words(index)	Range
字符集合	Characters(index)	Range
句子集合	Sentences(index)	Range
段落集合	Paragraphs(index)	Paragraph
节集合	Sections(index)	Section

从上面的表格中，我们知道：Words(index)、Characters(index)、Sentences(index)都将返回一个 Range 对象。Paragraphs 和 Sections 集合中的成员分别是 Paragraph 和 Section 对象。但 Range 属性（返回 Range 对象）对 Paragraph 和 Section 对象都是有效的，也可以使用 Range 对象的属性和方法。

下面的代码将活动文档的第一段复制到剪贴板上，并删除第一段的第一句。

ActiveDocument.Paragraphs(1).Range.Copy

ActiveDocument.Paragraphs(1).Range.Words(1).delete

下面的示例设置活动文档第一个单词的大小写。

ActiveDocument.Words(1).Case = wdUpperCase

下面的示例将当前节的下边距设置为 0.5 英寸。

Selection.Sections(1).PageSetup.BottomMargin = InchesToPoints(0.5)

下面的示例将活动文档的字符间距设为两倍（Content 属性返回一个 Range 对象）。

ActiveDocument.Content.ParagraphFormat.Space2

10.1 用 VBA 程序操作文档中的文本

10.1.1 选定文档中的文本

使用 Select 方法可以选定文档中的一项。Select 方法对许多对象都是可用的，例如 Bookmark、Field、Range 和 Table 对象等。

下面的示例选定活动文档中的第一张表格。

ActiveDocument.Tables(1).Select

下面的示例选定活动文档中的第一个域。

ActiveDocument.Fields(1).Select

下面的示例选定活动文档中的前四个段落。Range 方法用于创建一个 Range 对象，该对象引用前四个段落；然后将 Select 方法应用于 Range 对象。

```
Set myRange = ActiveDocument.Range( _
  Start:=ActiveDocument.Paragraphs(1).Range.Start, _
  End:=ActiveDocument.Paragraphs(4).Range.End)
myRange.Select
```

在选定文本之后，使用 Collapse 方法可以将文本的选定部分（Selection 或 Range 对象）折叠到开始或结束点的位置。

Collaps 方法的语法如下：

```
expression.Collapse(Direction)
```

上式中各项的意义如下：

expression：必需。该表达式返回一个 Range 或 Selection 对象。

Direction：Variant 类型，可选。代表折叠某一区域或所选内容的方向。其值如表 10-2 所示。

表 10-2　Direction 参数的可选值

Direction 参数值	**意义**
WdCollapseStart	将所选区域折叠到区域起始点，默认值
WdCollapseEnd	将所选区域折叠到区域结束点

下面的示例在所选内容的开头将所选内容折叠为一个插入点。

```
Selection.Collapse Direction:=wdCollapseStart
```

下面的示例将 myRange 对象折叠到结束位置（在第一个单词之后）。

```
Set myRange = ActiveDocument.Words(1)
myRange.Collapse Direction:=wdCollapseEnd
```

在选定文本之后，在 VBA 程序中还可对选定内容进行扩展。

下面的示例使用 MoveEnd 方法扩展选定内容的结尾以包含三个附加的单词。MoveLeft、MoveRight、MoveStart、MoveUp 和 MoveDown 方法也可以用来扩展 Selection 对象（具体内容可参考 Word2000 的帮助）。

```
Selection.MoveEnd Unit:=wdWord, Count:=3
```

下面的示例使用 MoveEnd 方法扩展 myrange，以包含活动文档的前三个段落。

```
Set myRange = ActiveDocument.Paragraphs(1).Range
myRange.MoveEnd Unit:=wdParagraph, Count:=2
```

10.1.2 对文本应用格式

对文档应用格式主要包括三部分内容，即字体格式，段落格式和页面的设置。

1. 设置字体的格式

通过使用 Font 对象，可以设置文本的字体格式。

Font 对象包含了各种字体属性（字体名称、字体大小、颜色等）。用 Font 属性可返回 Font 对象。

下例将活动文档首段的格式设置为 24 磅、Arial 和倾斜。

```
Set myRange = ActiveDocument.Paragraphs(1).Range
With myRange.Font
  .Bold = True
  .Name = "Arial"
  .Size = 24
End With
```

下例将活动文档中标题 2 的样式更改为 Arial 和加粗。

```
With ActiveDocument.Styles(wdStyleHeading2).Font
  .Name = "Arial"
  .Italic = True
End With
```

可用关键字“New”来创建一个独立应用的新 Font 对象。下例先创建一个 Font 对象，再设置一些格式属性，然后将该 Font 对象应用于活动文档的首段。

```
Set myFont = New Font
```

```
myFont.Bold = True
myFont.Name = "Arial"
ActiveDocument.Paragraphs(1).Range.Font = myFont
```

用 Duplicate 属性可复制 Font 对象。下列示例创建一个新的字符样式，该样式除具有选定内容中的字符格式外还具有倾斜格式。而选定内容的格式并没有改变。

```
Set aFont = Selection.Font.Duplicate
aFont.Italic = True
ActiveDocument.Styles.Add(Name:="Italics", _
   Type:=wdStyleTypeCharacter).Font = aFont
```

2. 设置段落的格式

通过使用 ParagraphFormat 对象可以设置文本中段落的格式。

ParagraphFormat 对象代表段落的所有格式。用 Format 属性返回一个或多个段落的 ParagraphFormat 对象; 用 ParagraphFormat 属性返回选定内容、范围、样式、Find 对象或 Replacement 对象的 ParagraphFormat 对象。

下例将活动文档第三段居中对齐。

```
ActiveDocument.Paragraphs(3).Format.Alignment = _
    WdAlignParagraphCenter
```

下例将选定段落的段前距离设置为 12 磅。

```
Selection.FormatParagraph.SpaceBefore = 12
```

可用 New 关键词创建新的单独的 ParagraphFormat 对象。下面示例创建一个 ParagraphFormat 对象，为其设置一些格式属性，然后将其所有属性应用于活动文档的第一个段落。

```
Dim myParaF As New ParagraphFormat
myParaF.Alignment = wdAlignParagraphCenter
myParaF.Borders.Enable = True
ActiveDocument.Paragraphs(1).Format = myParaF
```

也可用 Duplicate 属性生成原有的 ParagraphFormat 对象的单独副本。下面示例复制活动文档第一段落的段落格式并将格式保存于 myDup。然后将 myDup 的左缩进量更改为 1 英寸，创建新文档，将文字插入文档，并将 myDup 的段落格式应用于该文字。

```
Set myDup = ActiveDocument.Paragraphs(1).Format.Duplicate
myDup.LeftIndent = InchesToPoints(1)
Documents.Add
Selection.InsertAfter "This is a new paragraph."
Selection.Paragraphs.Format = myDup
```

3. 设置文本的样式

通过使用 Styles 对象，可以设置文本的样式。

Styles 对象代表一个内置或用户定义的样式。Style 对象是 Styles

集合的一个成员。Styles 集合包含了指定文档中所有的内置样式和用户自定义样式。

Style 对象将样式的属性（如字体、字体样式、段落间距等）表示为 Style 对象的属性。

使用 Style 属性可以返回一个 Sytle 对象。将通过使用 Range、Document 或者 Selcction 对象的 Style 属性，即可在范围、段落或多个段落中对文本应用样式。

下例将常用样式应用于活动文档的前四段。

```
Set myRange = ActiveDocument.Range( _
    Start:=ActiveDocument.Paragraphs(1).Range.Start, _
    End:=ActiveDocument.Paragraphs(4).Range.End)
myRange.Style = wdStyleNormal
```

下例将“标题 1”样式应用于选定内容的第一段。

```
Selection.Paragraphs(1).Style = wdStyleHeading1
```

下例生成一名为“Bolded”的字符样式并应用于选定内容。

```
Set myStyle = ActiveDocument.Styles.Add(Name:="Bolded", _
    Type:=wdStyleTypeCharacter)
myStyle.Font.Bold = True
Selection.Range.Style = "Bolded"
```

使用 Styles(index)也可以返回一个 Style 对象，其中 index 为

WdBuiltinStyle 常量（内置样式名）或索引号。样式名的拼写和间隔必须正确，而大小写则无关紧要。

样式索引号代表以字母顺序排列的样式名列表中该样式的位置。注意 Styles(1) 为列表中的第一个样式。下例显示 Styles 集合中第一个样式的基本样式和样式名称。

```
MsgBox "Base style= " _
    & ActiveDocument.Styles(1).BaseStyle & Visual BasicCr _
    & "Style name= " & ActiveDocument.Styles(1).NameLocal
```

下例更改活动文档中用户定义样式“Color”中的字体名称。

```
ActiveDocument.Styles("Color").Font.Name = "Arial"
```

以下示例将内置标题 1 样式设置为非黑体，并更新文档中的样式。

```
ActiveDocument.Styles(wdStyleHeading1).Font.Bold = False
ActiveDocument.UpdateStyles
```

4. 文档的页面设置

使用 PageSetup 对象，可以对文档的页面进行设置。

可用文档对象或者区域对象的 PageSetup 属性可以得到一个 PageSetup 对象。PageSetup 对象代表了对页面设置描述，它包含了文档的所有页面设置属性（左边距、下边距、纸张大小，等等）。

下例将活动文档的第一节设为横向并打印该文档。

```
ActiveDocument.Sections(1).PageSetup.Orientation = _
    wdOrientLandscape
ActiveDocument.PrintOut
```

下例设置“Sales.doc”文档的所有边距。

```
With Documents("Sales.doc").PageSetup
    .LeftMargin = InchesToPoints(0.75)
    .RightMargin = InchesToPoints(0.75)
    .TopMargin = InchesToPoints(1.5)
    .BottomMargin = InchesToPoints(1)
End With
```

本示例将页眉与页面顶端、页脚与页面底端的距离分别设置为 18 磅（0.25 英寸）。此操作有效范围为选定内容所在的节。

```
With Selection.PageSetup
    .FooterDistance = 18
    .HeaderDistance = 18
End With
```

10.1.3 在文档中处理文字

使用 InsertAfter 或 InsertBefore 方法可以在 Selection 或 Range 对象前后插入文字。InsertAfter 和 InsertBefore 方法的语法如下：

```
expression.InsertAfter(Text)
```

expression：必需。该表达式返回一个 Range 或 Selection 对象。

Text：必需。 String 类型，表达要插入的文本

下例在活动文档结尾处插入文字。

```
ActiveDocument.Content.InsertAfter Text:=" the end."
```

如果应用本方法的区域或所选内容是一个完整的段落，则在段落结束标记之后插入文本，插入文本将出现在下一段开头。如要在段尾插入文本，可先确定结束点，再从该位置减去 1（因段落标记是一个字符），如下例所示。

```
Set doc = ActiveDocument
Set rngRange = _
    doc.Range(doc.Paragraphs(1).Start, _
    doc.Paragraphs(1).End - 1)
rngRange.InsertAfter _
    " This is now the last sentence in paragraph one."
```

✍ Range 对象或 Selection 对象在使用了 InsertBefore 或 InsertAfter 方法之后，会扩展并包含新的文本。使用 Collapse 方法可以将 Selection 或 Range 折叠到开始或结束位置。

通过 InsertBefore 方法和 Chr 函数，可以插入引号、制表符和不

间断连字符等字符。也可以使用下列 Visual Basic 常量：Visual BasicCr（回车键）、Visual BasicLf（换行键）、Visual BasicCrLf（回车键＋换行键）和 Visual BasicTab（制表键）。

使用 Selection 对象的 TypeText 方法也可以在文档中插入指定的文本。如果 ReplaceSelection 属性为 True，则用指定文本替换所选内容。如果 ReplaceSelection 为 False，则在所选内容之前插入指定的文本。

✍ 改变 ReplaceSelection 属性，相当于按下 Insert 键。

TypeText 方法的语法如下：

```
expression.TypeText(Text)
```

expression：必需。该表达式返回一个 Selection 对象。

Text：String 类型，必需。代表要插入的文字

本示例删除插入点（折叠的所选内容）前面的字符。

```
With Selection
    .Collapse Direction:=wdCollapseEnd
    .TypeBackspace
End With
```

本示例将所选内容扩展到当前段落的末尾（包括段落标记），然后删除所选内容。

```
With Selection
    .EndOf Unit:=wdParagraph, Extend:=wdExtend
```

```
        .TypeBackspace
    End With
```

在处理文字时，我们还可以使用 Selection 对象的 TypeBackspace 方法和 TypeParagraph 方法，它们的功能分别相当于按下 BackSpace 键和回车键。

本示例将所选内容折叠至末尾，删除选定内容前的一个字符，然后在其后插入一个新段落。

```
    With Selection
      .Collapse Direction:=wdCollapseEnd
      .TypeBackSpace
           .TypeParagraph
    End With
```

Range 或者 Selection 对象的 Text 属性是一个可读写的“String”类型的属性，它返回指定区域或者所选内容中的文本。通过对 Text 属性的修改，可以对指定区域或者所选内容中的文本进行编辑。

本示例用“Dear”替换当前文档的第一个词。

```
    Set myRange = ActiveDocument.Words(1)
    myRange.Text = "Dear "
```

本示例用“Goodbye”替换活动文档中的“Hello”。

```
    Set myRange = ActiveDocument.Content
```

```
With myRange.Find
    .ClearFormatting
    .Replacement.ClearFormatting
    .Text = "Hello"
    .Replacement.Text = "Goodbye"
    .Execute Replace:=wdReplaceAll
End With
```

下面的示例在当前文档的上部添加单词 Title 。第一段居中对齐，并在该段落之后添加了半英寸的间距。单词 Title 的格式设为 24 磅 Arial 字体。

Set oRange = ActiveDocument.Range(Start:=0, End:=0)

```
With oRange
    .InsertAfter Text:="Title"
    .InsertParagraphAfter
    .Font.Name = "Arial"
    .Font.Size = 24
End With
With ActiveDocument.Paragraphs(1)
    .Alignment = wdAlignParagraphCenter
    .SpaceAfter = InchesToPoints(.5)
```

End With

下面的示例在选定内容的第一段格式前切换间距。宏会检索取值前的间距，如果值是 12 磅，则删除格式前的间距（SpaceBefore 属性设为零)。如果段前间距不是 12，则将 SpaceBefore 属性设为 12 磅。

```
Set oParagraph = Selection.Paragraphs(1)
    If oParagraph.SpaceBefore = 12 Then
        oParagraph.SpaceBefore = 0
    Else
        oParagraph.SpaceBefore = 12
End If
```

下面的示例切换选定文本的加粗格式。

```
Selection.Font.Bold = wdToggle
```

将左边距增加 0.5 英寸

下面的示例将左边距增加 0.5 英寸。PageSetup 对象包含文档的所有页面设置属性（左边距、下边距、纸张大小等）。LeftMargin 属性用于返回并设置左边距。

```
iMargin = ActiveDocument.PageSetup.LeftMargin
iMargin = iMargin + InchesToPoints(0.5)
ActiveDocument.PageSetup.LeftMargin = iMargin
```

10.1.4　查找并替换文档中的文本或格式

在 VBA 程序中可以执行对文档中文本和格式的查找和替换功能。要使用这些功能，首先应该了解 Find 对象和 Replace 对象。

Find 对象代表查找操作的执行条件。**Find** 对象的属性和方法与“替换”对话框中的选项一致。可用 Selection 或者 Range 对象的 Find 属性可以返回一个 Find 对象。

Find 对象的主要的属性和方法有：

- Wrap 属性

用 Wrap 属性可以指定当不从文档开头开始进行搜索而又到达文档底部时（或者如果 Forward 设置为 False 或相反），或在指定的选定内容或区域中未找到搜索文字时，返回或设置需要进行的操作。Long 类型，可读写。

Wrap 属性的可选值如表 10-3 所示。

表 10-3　Wrap 属性的可选值

常量	描述
WdFindAsk	在搜索完所选内容或范围之后，Word 显示一条消息，询问是否搜索文档的其他部分。
WdFindContinue	当到达搜索范围的开始或末尾时，继续进行查找操作。
WdFindStop	当到达搜索范围的开始或末尾时，停止查找操作。

- Forward 属性

在文档中，如果查找操作向前搜索，则 Forward 属性为 True。如果向后搜索，则本属性为 False。Forward 属性是一个可读写的 Boolean 类型属性。

- Found 属性

Found 属性是一个可读写的 Boolean 类型属性。在查找过程中，如果查找到了匹配项，则 Found 属性的值为 True，否则为 False。

下例的功能是：在查找选定部分之前，清除查找条件中的格式。如果查找到的单词“Hello”为加粗格式，则选定整个段落，并将其复制至剪贴板上。

```
With Selection.Find
    .ClearFormatting
    .Font.Bold = True
    .Execute FindText:="Hello", Format:=True, Forward:=True
    If .Found = True Then
        .Parent.Expand Unit:=wdParagraph
        .Parent.Copy
    End If
End With
```

- MatchCase 属性

如果 MatchCase 属性为 True，则查找过程中区分大小写。Boolean

类型，可读写。

● MatchByte 属性

如果 MatchByte 属性为 True，则查找过程中区分全角和半角。Boolean 类型，可读写。

● Text 属性

Text 属性为 Boolean 类型，可读写。 代表需要查找的文字。

● ClearFormatting 方法

取消任何为进行查找或替换操作所指定的格式。相应于“编辑”菜单“查找和替换”对话框的“不限定格式”按钮。

如果要确保在查找或替换操作的条件中不包含不需要的格式，可在进行该操作前使用本方法。

ClearFormatting 方法的语法如下：

```
expression.ClearFormatting
```

其中 expression：必需。该表达式返回 Find 或 Replacement 对象

本示例在将整个活动文档中的单词“Inc.”替换为“incorporated”之前，清除格式限定。

```
Set myRange = ActiveDocument.Content
With myRange.Find
    .ClearFormatting
    .Replacement.ClearFormatting
```

```
        .MatchWholeWord = True
        .Execute FindText:="Inc.", _
            ReplaceWith:="incorporated", Replace:=wdReplaceAll
    End With
```

- Execute 方法

Find 对象的 Execute 方法用于运行指定的查找操作。如果查找成功，则返回 True。

Execute 方法的语法如下：

expression.Execute(FindText, MatchCase, MatchWholeWord, MatchWildcards, MatchSoundsLike, MatchAllWordForms, Forward, Wrap, Format, ReplaceWith, Replace, , MatchControl)

上式中各项的意义分别如下：

expression ： 必需。该表达式返回 Find 对象。

FindText： Variant 类型，可选。指定需查找的文本。可用空字符串 ("") 查找格式。也可通过指定适当的字符代码查找特殊字符。例如，“^p”对应段落标记，“^t”对应制表符。如果需要使用特殊字符列表，请参阅查找替换特殊字符和文档元素示例。

如果 MatchWildcards 为 True，则可以指定通配符及其他高级搜索条件。例如，“*(ing)” 将查找以“ing”结尾的所有单词。。

若要搜索符号字符，可键入 (^) 字符，零（0），然后键入符号字

符的代码。例如，“^0151”对应一条长划线（—）。

MatchCase：　　Variant 类型，可选。如果是 True，则查找文本需区分大小写。相当于“编辑”菜单“查找和替换”对话框中的“区分大小写”复选框。

MatchWholeWord：Variant 类型，可选。如果为 True，则只查找匹配的完整单词，而并非作为一个长单词的一部分的文字。相当于“编辑”菜单“查找和替换”对话框中的“全字匹配”复选框。

MatchWildcards：　Variant 类型，可选。如果为 True，则查找的文字包含特殊搜索操作符。相当于“编辑”菜单“查找和替换”对话框中的“使用通配符”复选框。

MatchAllWordForms：　Variant 类型，可选。如果为 True，则查找文字的所有形式（例如，“ sit” 将包含“sitting”和“sat”）。相当于“编辑”菜单“查找和替换”对话框中的“查找单词的各种形式”复选框。

Forward：Variant 类型，可选。如果为 True，则向下（向文档尾部）搜索。

Wrap：　　　Variant 类型，可选。如果搜索从不是文档开头的位置开始，并到达文档末尾（如 Forward 设置为 False，则相反），用本参数控制接下来的操作。当在选定内容或区域中没有找到搜索文字时，本参数也控制接下来的操作。

Format:　　　Variant 类型，可选。 如果为 True，则查找格式而

非文字。

ReplaceWith: Variant 类型，可选。替换文字。要删除由 Find 参数指定的文字，可使用空字符串 ("")。与 Find 参数相似，本参数也可以指定特殊的字符和高级搜索条件。要将图形对象或者其他非文本项指定为替换内容，可将这些项目置于“剪贴板”上，然后将 ReplaceWith 指定为“ ^c”。

Replace: Variant 类型，可选。指定执行替换的个数：一个、全部或者不替换。可为下列 WdReplace 常量之一：wdReplaceAll、wdReplaceNone 或 wdReplaceOne。

本示例查找并选定下次出现的“library”。

```
With Selection.Find
    .ClearFormatting
    .MatchWholeWord = True
    .MatchCase = False
    .Execute FindText:="library"
End With
```

本示例在活动文档中查找所有的“hi”，并且将其替换为“hello”。

```
Set myRange = ActiveDocument.Content
myRange.Find.Execute FindText:="hi", _
    ReplaceWith:="hello", Replace:=wdReplaceAll
```

下例查找和选定下一个出现的“hi”。

```
With Selection.Find
    .ClearFormatting
    .Text = "hi"
    .Execute Forward:=True
End With
```

Replacement 对象代表查找-替换操作的替换条件。它的属性和方法与“替换”对话框中的选项一致。用 Find 对象的 Replacement 属性可以返回一个 Replacement 对象。

下例在活动文档中查找所有“hi”并将其替换为“hello”。

```
Set myRange = ActiveDocument.Content
myRange.Find.Execute FindText:="hi", ReplaceWith:="hello", _
    Replace:=wdReplaceAll
```

如果在执行查找操作时，将要查找文字和要替换文字设为空字符串（""），并把 Execute 方法的 Format 参数设为 True，则可以查找和替换格式设置。

下例删除活动文档中的所有加粗格式。在 Find 对象中 Bold 属性设为 True，而在 Replacement 对象中该属性设为 False。

```
Sub ReplaceFormat()
    With ActiveDocument.Content.Find
```

```
.ClearFormatting
.Font.Bold = True
.Text = ""
With .Replacement
    .ClearFormatting
    .Font.Bold = False
    .Text = ""
End With
.Execute Format:=True, Replace:=wdReplaceAll
End With
End Sub
```

在 Selection 对象中使用 Find 对象时，如果找不到找到符合条件的文本，则 Selection 对象不会改变，但如果找到了符合条件的文本，选定内容将会改变。例如，在下面的语句中，选定部分将变为搜索到的字符串“blue”。

```
Selection.Find.Execute FindText:="blue", Forward:=True
```

但在使用 Selection 对象中调用 Range 对象的 Find 方法时，找到符合选择条件的文本后选定内容不会改变，但 Range 对象将会重新定义。

下例定位活动文档中第一次出现的“blue”。如果在文档中找到

“blue”，则重新定义 myRange 并将“blue”的格式设为粗体。

```
Set myRange = ActiveDocument.Content
myRange.Find.Execute FindText:="blue", Forward:=True
If myRange.Find.Found = True Then myRange.Bold = True
```

✍ 注意：除非另外指定，否则替换文本将沿用文档中被替换文本的格式。例如，如果用“xyz”替换“abc”，那么粗体“abc”将被粗体字符串“xyz”所替换。同样，如果 MatchCase 为 False，那么查找到的大写文本将被大写的替换文本替换，而无论搜索文本和替换文本是否大小写。上例中，“ABC”将被“XYZ”替换。

10.2　在 VBA 程序中使用 Word 中的表格

要学习在程序中操纵 Word 中的表格，首先应该了解 Table 对象和 Tables 对象集合。

10.2.1　Tables 对象集合和 Table 对象

Tables 对象集合是由 Table 对象组成的集合，它代表选定内容、范围或文档中的所有表格。

使用 Tables 属性可以返回一个 Tables 集合。下面的语句将在对话框中显示活动文档中的表格数目。

Msgbox ActiveDocument.Tables.Count

使用 Tables 对象的 Add 方法可以在指定范围内新增一表格。Add 方法的语法如下：

expression.Add(Range, NumRows, NumColumns, DefaultTableBehavior, AutoFitBehavior)

上式中各项的意义分别如下：

expression ： 必需。整个 Add 表达式返回一个 Tables 对象。

Range：Range 类型，必需。指定表格放置的区域。如果该区域未折叠，表格将取代该区域。

NumRows：Long 类型，必需。指定要插入表格的行数。

NumColumns：Long 类型，必需。指定要插入表格的列数。

DefaultTableBehavior：Variant 类型，可选。该属性指定 Microsoft Word 是否要根据单元格中的内容自动调整表格单元格的大小（“自动调整”功能）。其值可取下列常量之一：wdWord8TableBehavior（取消“自动调整”功能）或 wdWord9TableBehavior（启用“自动调整”功能）。默认值为 wdWord8TableBehavior。

AutoFitBehavior：Variant 类型，可选。该属性用于设置 Word 调整表格大小的“自动调整”规则。其值可取以下 WdAutoFitBehavior 常量之一：wdAutoFitContent、wdAutoFitFixed 或 wdAutoFitWindow。如果 DefaultTableBehavior 属性设置为 wdWord8TableBehavior，将忽

略该参数。

下例的功能是用一个 3 x 4 表格代替文档的选定部分。

```
Set myRange = ActiveDocument.Selection.Range
ActiveDocument.Tables.Add Range:=myRange, NumRows:=3, NumColumns:=4
```

Table 对象代表一个表格，它是 Tables 对象集合的一个成员。

在前面我们已经讲过，使用 Tables 集合的 Add 方法可以返回一个 Table 对象。此外用 Tables(index)也可以得到 Table 对象，其中 index 为索引号，代表选定内容、范围或文档中表格的编号。

下例将活动文档中的第一个表格转换为文本。

```
ActiveDocument.Tables(1).ConvertToText Separator:=wdSeparateByTabs
```

使用 Table 对象的 Split 方法可以拆分表格。Split 方法的语法如下：

```
expression.Split(BeforeRow)
```

上式中各项的意义分别如下：

expression： 该表达式返回一个 Table 对象。

BeforeRow： Variant 类型，必需。代表要在哪一行对表格进行拆分，其值可以是一个 Row 对象或某一行的行号。例如，如果要在第二行和第三行之间拆分表格，则 BeforeRow 的值应为 3。

本示例在活动文档中创建一张 5 x 5 的表格，并且在第三行之前

进行拆分。Set newDoc

```
Sub Example()
    Set myTable = ActiveDocument.Tables.Add(Range:=Selection.Range, _
        NumColumns:=5, NumRows:=5)
    myTable.Split(BeforeRow:=myTable.Rows(3))
End Sub
```

10.2.2 引用表格的行和列

使用 Table 对象的 Rows 属性，可以返回一个 Rows 集合。该集合代表了指定的选定部分、区域或表格中的所有表格行。

例如，下面的语句居中对齐活动文档第一张表格的各行。

```
ActiveDocument.Tables(1).Rows.Alignment = wdAlignRowCenter
```

要引用表格中的某一行，可以用 Rows(index) 可返回单个的 Row 对象。其中的 index 是索引序号。代表该行在选定部分、区域或表格中的位置。

下列示例删除活动文档中第一张表格的首行。

```
ActiveDocument.Tables(1).Rows(1).Delete
```

同样，用 Columns 属性可以引用表格中的所有列，用 Columns(index)

可以引用表格中的某一列。

10.2.3 引用表格单元

要引用表格中所有的单元格，可以使用 Cells 属性。Cells 属性可以应用于 Selection 对象、Range 对象、Row 对象和 Column 对象，值得注意的是：用 Table 对象不能直接调用 Cells 属性。

下例的功能是：创建一个 3×3 表格并给表格中的每个单元格分配一个顺序单元格编号。注意 Cells 属性是如何调用的。

```
Sub Example()
    Set newDoc = Documents.Add
    Set myTable = newDoc.Tables.Add(Selection.Range,3,3)
    i = 1
    For Each c In myTable.Range.Cells
        c.Range.InsertAfter "Cell " & i
        i = i + 1
    Next c
End Sub
```

使用 Table 对象的 Cell 方法可以使用引用表格中的一个单元格。Cell 方法的语法如下：

```
expression.Cell(Row, Column)
```

上式中各项的意义分别为：

expression：必需。代表一个 Table 对象。

Row：Long 类型，必需。代表返回的表格行数。可以是介于 1 和表格行数之间的任意整数。

Column：Long 类型，必需。代表返回的表格单元格数目。可以是介于 1 和表格列数之间的任意整数。

本示例在新文档中创建一个 3×3 表格，并在表格的第一个和最后一个单元格中插入文本。

```
Set newDoc = Documents.Add
Set myTable = newDoc.Tables.Add(Selection.Range, 3, 3)
With myTable
    .Cell(1,1).Range.InsertAfter "First cell"
    .Cell(myTable.Rows.Count, _
        myTable.Columns.Count).Range.InsertAfter "Last Cell"
End With
```

使用 Cell 对象或 Cells 集合的 Merge 和 Split 方法可以拆分单元格。例如：

下面的语句将第一张表格的第一个单元格拆分为两个单元格。

```
ActiveDocument.Tables(1).Cell(1, 1).Split NumColumns:=2
```

下面的语句首先将选定单元格合并为一个，再将其拆分为同一行

上的三个单元格。

```
If Selection.Information(wdWithInTable) = True Then
    Selection.Cells.Split NumRows:=1, NumColumns:=3, _
        MergeBeforeSplit:= True
End If
```

10.2.4　创建表格、插入文字

下面的示例在文档的开头插入一张 3 行 4 列的表格，并向每个单元格中添加文字。

```
Sub Example()
  Set oDoc = ActiveDocument
  Set oTable = oDoc.Tables.Add( _
      Range:=oDoc.Range(Start:=0, End:=0), NumRows:=3, _
      NumColumns:=4)
  iCount = 1
  For Each oCell In oTable.Range.Cells
      oCell.Range.InsertAfter "Cell " & iCount
      iCount = iCount + 1
  Next oCell
  oTable.AutoFormat Format:=wdTableFormatColorful2, _
```

```
        ApplyBorders:=True, ApplyFont:=True, ApplyColor:=True
    End Sub
```

从上面的示例中可以看到：For Each...Next 结构可以用来循环遍历表格中的每个单元格。用 InsertAfter 方法可以向表格单元格（Cell 1、Cell 2 等等）添加文字。

要删除表格单元格中的内容，可以使用 Cell 对象的 Delete 方法。

下面的语句用于删除表格中第一行、第一列的单元格中的文字。

```
If ActiveDocument.Tables.Count >= 1 Then
    With ActiveDocument.Tables(1).Cell(Row:=1, Column:=1).Range
        .Delete
    End With
End If
```

10.2.5 设置表格的格式

使用 AutoFormat 方法可以设置表格的格式。AutoFormat 方法的语法如下：

```
expression.AutoFormat(Format, ApplyBorders, ApplyShading, ApplyFont, ApplyColor,, AutoFit)
```

上式中各项的意义分别如下：

expression：必需。代表一个 Table 对象。

Format：Variant 类型，可选。代表预定义的表格格式。可选 WdTableFormat 常量之一。默认值为 wdTableFormatSimple1（简明 1 型）。

ApplyBorders ：Variant 类型，可选。如果为 True，则应用指定格式的边框属性。默认值为 True。

ApplyShading：Variant 类型，可选。如果为 True，则应用指定格式的底纹属性。默认值为 True。

ApplyFont：Variant 类型，可选。如果为 True，则应用指定格式的字体属性。默认值为 True。

ApplyColor：Variant 类型，可选。如果为 True，则应用指定格式的颜色属性。默认值为 True。

AutoFit：Variant 类型，可选。如果为 True，则在不改变单元格内文字换行情况下尽量缩小表格列宽。默认值为 True。

下例在新文档中创建一个 5×5 的表格，并将“彩色型 2”格式的所有属性应用于此表格。

```
Sub Example()
  Set newDoc = Documents.Add
  Set myTable = newDoc.Tables.Add(Range:=Selection.Range, _
        NumRows:=5, NumColumns:=5)
```

```
myTable.AutoFormat Format:=wdTableFormatColorful2
End Sub
```

此外，用 Table 对象的 Borders 属性和 Shading 属性还可以设置表格的边框和底纹。

下例给文档中每个表格的首行设置竖线底纹，并在四周添加艺术型边框。

```
Sub Example()
    For Each aBorder In ActiveDocument.Sections(1).Borders
        With aBorder
            .ArtStyle = wdArtSeattle
            .ArtWidth = 20
        .Rows(1).Shading.Texture = _
        wdTextureVertical
        End With
    Next aBorder
End Sub
```

10.2.6 表格与文字的相互转换

用 ConvertToTable 方法可以将文字转换为表格。ConvertToTable 方法可以被一个 Range 对象或 Selection 对象调用。

有关 ConvertToTable 方法的语法，请参照 Word 2000 的帮助

本示例在插入点插入文字，并将以逗号为分隔符的文本转换为格式化的表格。

```
Sub Example()
  With Selection
      .Collapse
      .InsertBefore "one, two, three"
      .InsertParagraphAfter
      .InsertAfter "one, two, three"
      .InsertParagraphAfter
  End With
  Set myTable = _

Selection.ConvertToTable(Separator:=wdSeparateByCommas, _
      Format:=wdTableFormatList8)
End Sub
```

同样，用 Table 对象的 ConvertToText 方法，可以将表格转换为文字。

用 ConvertToText 方法可以将表格转换为文本并返回一个 Range 对象，该对象代表转换得到的带分隔符的文本。

ConvertToText 方法的语法如下：

expression.ConvertToText(Separator, NestedTables)

expression： 必需。代表一个 Row、Rows 或 Table 对象。

Separator： Variant 类型，可选。用以分隔被转换列的分隔符（被转换行由段落标记分隔）。可取下列 WdTableFieldSeparator 常量之一：wdSeparateByCommas、wdSeparateByDefaultListSeparator、wdSeparateByParagraphs 或 wdSeparateByTabs。默认值为 wdSeparateByTabs。

NestedTables：Variant 类型，可选。如果为 True，则将嵌套的表格转换为文本。如果 Separator 不是 wdSeparateByParagraphs，则此参数将被忽略。其默认值为 True

下例将包含选定内容的表格转换为文本，各列之间用空格分隔。

```
Sub ToText()
 If Selection.Information(wdWithInTable) = True Then
     Selection.Tables (1). ConvertToText Separator:=" "
 Else
     MsgBox "The insertion point is not in a table."
 End If
End Sub
```

10.3　检查文档中的拼写和语法

要对一篇文档或者一个文档区域进行拼写和语法的检查，可以使用 Document 或者 Range 对象的 CheckGrammer 和 CheckSpelling 方法。这两种方法的区别在于：CheckSpelling 方法只检查文档中的拼写错误，而 CheckGrammer 方法既检查文档中的拼写错误，也检查文档中的语法错误。

CheckSpelling 方法用于对开始对指定的文档或区域进行拼写检查。如果该文档或区域中有错误，本方法将显示“工具”菜单中的“拼写和语法”对话框，同时清除“语法检查”复选框。对于文档，本方法将检查所有有效的文档组成部分（比如标题、脚注和文本框）。CheckSpelling 方法的语法如下：

expression.CheckSpelling(CustomDictionary, IgnoreUppercase, AlwaysSuggest, CustomDictionary2 – CustomDictionary10)

expression：　必需。代表一个 Document 或 Range 对象。

CustomDictionary：　Variant 类型，可选。或为返回 Dictionary 对象的表达式，或为自定义词典的文件名。

IgnoreUppercase：　　Variant 类型，可选。如果值为 True，则忽略词语的大小写。

AlwaysSuggest：　　Variant 类型，可选。如果值为 True，则 Word

始终建议使用的拼写。

MainDictionary： Variant 类型，可选。或为返回 Dictionary 对象的表达式，或为主词典的文件名。

CustomDictionary2 ~ CustomDictionary10：Variant 类型 ，可选。或为返回 Dictionary 对象的表达式，或为一个自定义附加词典的文件名。在 Word 中，最多可以定义 9 个附加词典。

本示例开始对活动文档的所有有效组成部分进行拼写检查。

ActiveDocument.CheckSpelling

用 CheckSpelling 方法还可以对某个单独的字符串进行拼写检查，此时 CheckSpelling 方法被 Application 对象调用。其语法如下：

expression.CheckSpelling(Word, CustomDictionary, IgnoreUppercase, MainDictionary, CustomDictionary2 – CustomDictionary10)

上式中各项的意义如下

expression： 代表返回一个 Application 对象，可选。

Word： 必需。String 类型。要进行拼写检查的字符串。

下面的语句对所选的内容进行拼写检查。

pass = Application.CheckSpelling(Word:=Selection.Text)

MsgBox "Selection has no spelling errors: " & pass

CheckGrammer 方法可以在指定文档或区域进行拼写和语法检查。如果文档或区域有错误，此方法在“工具 ”菜单的“拼写和语法”对话框

中显示发现的错误，同时自动选中“检查语法”复选框。当此方法应用于文档时，它将检查文档中所有有效文字的语法（包括页眉、页脚和文本框）。

CheckGrammer 方法的语法如下：

```
expression.CheckGrammar()
```

下面的语句检查 MyDocument.doc 中第二节的拼写和语法。

```
Set Range2 = ActiveDocument.Sections(1).Range
Range2.CheckGrammar
```

可以对某个字符串进行拼写检查，此时应调用 Application 对象的 CheckGrammer 方法，其语法如下：

```
expression.CheckGrammar(String)
```

上式中各项的意义分别为：

expression： 必需。代表一个 Application 对象。

String： String 类型，必需。指需要进行语法检查的字符串。

下面的语句显示选定部分拼写和语法检查的结果。

```
pass = Application.CheckGrammar(String:=Selection.Text)
MsgBox "Selection is grammatically correct: " & pass
```

10.4　使用自动图文集

10.4.1　插入或删除自动图文集的词条

在 Word 中，AutoTextEntries 集合对应“自动图文集”词条列表。该列表与“工具”菜单上“自动更正”对话框中“自动更正”选项卡上的“自动更正”词条列表是对应的。

要在自动图文集中插入一个词条，可以使用 AutotextEntries 集合的 Add 方法和 AddRichText 方法。

Add 方法用于向“自动图文集”词条列表中加入一条纯文本的词条。其语法如下：

expression.Add(Name, Value)

上式中各项的意义分别为：

expression：　必需。该表达式返回一个 Autotextentries 对象。

Name：　String 类型，必需。将自动替换为由 Value 所指定的文字。

Value：　String 类型，必需。将自动替换由 Name 所指定的文字。

下面的语句将新建一个自动图文集词条，其名称为“VISUAL BASICE”，更正值为“Visual Basic 编辑器”。

AutoCorectEntries.Add　Name:=“VISUAL BASICE”,Value=

“Visual Basic 编辑器”

用 AddRichText 方法可以创建带格式的“自动更正”词条，保留选定范围的所有文本属性。如果未使用 AddRichText 方法，则添加的“自动更正”词条将设置为当前样式。

AddRichText 方法的语法如下：

expression.AddRichText(Name, Range)

上式中各项的意义分别如下：

expression：必需。该表达式返回一个 Autotextentries 对象。

Name：String 类型，必需。代表将为 Range 对象所自动替换的文本。

Range：Range 对象，必需。指在键入 Name 时 Word 自动插入的带格式的文本。

下面的在 Normal 模板中添加一个名为“Temp”的自动图文集词条，然后在一个消息框中显示该自动图文集词条的内容（文档中的第一个单词）。

```
Set myEntry = NormalTemplate.AutoTextEntries.Add(Name:="Temp", _
    Range:=ActiveDocument.Words(1))
MsgBox myEntry.Value
```

本示例将选定部分的内容存为附加模板中一个名为“Address”的自

动图文集词条。

```
If Len(Selection.Text) > 1 Then
    ActiveDocument.AttachedTemplate.AutoTextEntries.Add _
        Range:=Selection.Range, Name:="Address"
End If
```

要在“自动图文集”词条列表中删除某一个词条，可以使用Autotextentry对象的Delete方法 。

本示例实现的功能是：如果在附加模板中存在名为“Hello”的“自动图文集”词条，则将其删除。

```
For Each entry In
ActiveDocument.AttachedTemplate.AutoTextEntries
    If entry.Name = "Hello" Then entry.Delete
Next entry
```

✍ 用 Template 对象（模板）的 AutoTextEntries 属性可以得到 Autotextentries 集合，用 Autotextentries(index)可以得到一个 Autotextentry 对象。

10.4.2　在文档中插入自动图文集词条

AutoTextEntry 对象的 Insert 方法的功能是：在文档中的输入点处插入“自动图文集”词条，如果文档中有选定部分，则自动图文集词条

将替换文档的选定部分。如果不想替换指定范围，在使用本方法之前可使用 Collapse 方法，自动图文集词条将插入到选定部分之后。

Insert 方法的语法如下：

```
expression.Insert(Where, RichText)
```

上式中各项的意义分别为：

Where：Range 对象，必需。“自动图文集”词条的位置。

RichText：Variant 类型，可选。如果本参数为 True，则插入“自动图文集”词条时，应用原来的格式。

下面的语句将在所选内容的后面插入带格式的“自动图文集”词条“Sigh”。

```
Selection.Collapse Direction:=wdCollapseEnd
ActiveDocument.AttachedTemplate.AutoTextEntries("one").Insert
_
    Where:=Selection.Range, RichText:=True
```

10.4.3 将文档中的文字替换为自动图文集词条

在文档中，如果指定区域中的文字或某一区域周围的文字与现有的“自动图文集”词条名称相匹配的话，那么使用 InsertAutoText 方法将插入该“自动图文集”词条并替换原文字。如果没有找到匹配文字，则使用 InsertAutoText 方法会发生错误。

InsertAutoText 方法的语法如下：

```
expression.InsertAutoText
```

上式中的 expression 代表一个 Range 对象。

例如，在上面我们建立了一个“自动图文集”词条，其名称为“VISUAL BASICE”，更正值为“Visual Basic 编辑器”，那么，下面的语句将把输入的“VISUAL BASICE”替换为“Visual Basic 编辑器”.

```
Selection.TypeText "VISUAL BASICE"
Selection.Range.InsertAutoText
```

第十一章　操作磁盘和文件

在 VBA 程序中，我们常常要用到一些有关文档的信息和对文档的操作。本章将主要介绍如何通过 VBA 来查找文档并得到文档的属性。

11.1　查找和替换文档

通过使用 Office 2000 的 FileSearch 对象，可以在 VBA 程序中实现在给定的条件下查找文档。

下面首先简介一下 FileSearch 对象。

FileSearch 对象本身代表“文件”菜单中“打开”对话框的功能，它的父对象是 Application 对象。FileSearch 对象的下属对象和集合的结构图如图 11-1 所示。

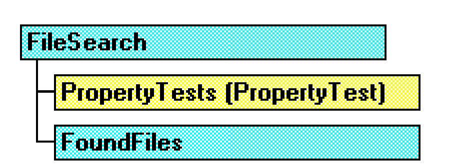


图 11-1　FileSearch 对象的结构图

从结构图中可见：FileSearch 对象下包含了 PorpertyTests 对象集合和 FoundFiles 对象。PropertyTests 对象集合代表的是搜索文件的条件，FoundFiles 对象代表的是符合搜索条件的文件。

使用 Application 对象的 FileSearch 属性可以得到一个 FileSearch 对

象。使用 FileSearch 对象的 PorpertyTests 属性和 FoundFiles 对象集合可以分别得到 FileSearch 对象集合和 FoundFiles 对象。

11.1.1 如何进行文件的搜索

要对文档进行搜索，必须使用 FileSearch 对象的 Execute 方法。

用 Execute 方法可以启动一次搜索，它的语法格式如下：

expression.Execute(SortBy, SortOrder, AlwaysAccurate)

上式中各项的意义分别如下：

expression：必需，代表 一个 FileSearch 对象。整个 Execute 方法返回查找到文件的数目。

SortBy：Variant 类型，可选。用来设置返回的文件的排序方法。SortBy 参数值可设置的值如表 11-1 所示。

表 11-1 SortBy 参数的可选值

SortBy 参数值	**意义**
MsoSortbyFileName	将搜索到的文件按文件名排列
MsoSortbyFileType	按文件类型排列
msoSortbyLastModif ied	按上次修改事件排列
MsoSortbySize	按文件大小排列

SortOrder： Variant 类型，可选。用于设置返回文件进行排序所用的顺序。该参数的可选值如图 11-2 所示。

表 11-2 SortOrder 参数的可选值

SortOrder 参数值	**意义**

MsoSortOrderAscending	升序排列
MsoSortOrderDescending	降序排列

AlwaysAccurate： Boolean 类型，可选。如果此参数值为 True，那么文件查找的范围包括文件索引最近一次更新后添加、修改或删除的文件。默认值为 True。

在下例中，我们搜索了当前目录下的所有文件并将搜索结果输入到一个新文件“SearchResult.doc”中，搜索结果按照文件类型排序。

```
Sub Searchfile()
    Set fs = Application.FileSearch.NewSearch
        If fs.Execute(SortBy:=msoSortByFileType)>0    Then
            Set newdoc = Documents.Add
            newdoc.Activate
            For i = 1 To fs.FoundFiles.Count
                    Selection.TypeText Text:=fs.FoundFiles.Item(i)
                    Selection.TypeParagraph
            Next
        Else
                MsgBox "No such files found"
        End If
    newdoc.SaveAs ("Search Results")
```

End Sub

11.1.2 使用默认的搜索条件

在上面的代码中，我们使用了 FileSearch 对象的 NewSearch 方法，如下：

```
Set fs = Application.FileSearch.NewSearch
```

NewSearch 方法的作用是将所有当前搜索条件重置为默认设置，即使得所有属性值在每次查找过程后仍然保持不变，用 NewSearch 方法可有选择地为下一次文件查找过程设置属性，而无须手动重置原先的属性值。其语法如下：

expression.NewSearch

上式中各项的意义如下：

expression： 不可省略。代表一个 FileSearch 对象。整个表达式返回一个 FileSearch 对象。

- FileSearch 方法可用于改变当前的搜索条件，但方法不会重置 LookIn 属性的值

11.1.3 搜索指定的目录

用 FileSearch 对象的 LookIn 属性，可以设定搜索的目录。

LookIn 是一个“String”类型的属性，可读写。LookIn 属性可用于返回或设置在指定的文件搜索过程中要搜索的文件夹。

如果我们不设置 FileSearch 对象的 LookIn 属性，则在指定的文件搜索过程中搜索当前的活动文件夹。

在指定要搜索的目录以后，如果需要同时搜索该目录中的所有子目录，则应将 SearchSubFolders 属性的值设置为 True；如果不需要搜索子目录，则应将该属性的值设为 False。

在下例中，我们在“C:\My Documents”目录（不包含子目录）中搜索所有符合条件的文档，并将搜索的结果存入文件“Search Results.doc”中。

```
Sub Searchfile()
    Set fs = Application.FileSearch
    With fs
    .NewSearch
    .LookIn = “C:\My documents”
    .SearchSubFolders = False
        If .Execute(SortBy:=msoSortByFileType)>0  Then
            Set newdoc = Documents.Add
            newdoc.Activate
            For i = 1 To .FoundFiles.Count
                    Selection.TypeText Text:=.FoundFiles.Item(i)
                    Selection.TypeParagraph
```

```
                Next
            Else
                    MsgBox "No such files found"
            End If
        End with
        newdoc.SaveAs ("Search Results")
    End Sub
```

11.1.4　搜索指定的文件名和文件类型

通过设置 FileSearch 对象的 FileName 属性，可以为文件搜索过程指定要搜索的文件名。FileName 属性是一个“String”类型的可读写属性。用它可返回或设置文件搜索过程中要查找的文件名。

在要搜索的文件名中可以包含 *（星号）或 ?（问号）通配符。问号通配符可以匹配任意一个单个字符。如键入“gr?y”可以匹配“gray”和“grey”。星号通配符可以匹配任意个字符。如键入“*.txt”可以查找到所有带 .TXT 扩展名的文件。

通过设置 FileSearch 对象的 FileType 属性，可以为文件搜索过程指定要搜索的文件类型。FileType 属性是一个“Variant”类型的可读写属性，用于返回或设置文件搜索过程中要查找的文件类型。

FileType 的属性值如表 11-3 所示：

表 11-3　FileType 属性的可选值

FileType 参数值	**意义**
MsoFileTypeAllFiles	搜索所有文件
MsoFileTypeBinders	活页夹文件
MsoFileTypeDatabases	数据库文件
MsoFileTypeExcelWorkbooks	Excel 文件
MsoFileTypeOfficeFiles	Office 文件
MsoFileTypePowerPointPresentations	PowerPoint 文件
MsoFileTypeTemplates	模板文件
MsoFileTypeWordDocuments	Word 文档

- FileType 属性的默认值为 msoFileTypeOfficeFiles，即用 Office 可以打开的文件。此时文件类型包含以下任意扩展名的文件: *.doc、*.xls、*.ppt、*.pps、* obd、*.mdb、*.mpd、*.dot、*.xlt、*.pot、*.obt、*.htm 或 *.html

在下面的例子中，我们对上面的代码作进一步改进，在“C:\My Documents”目录中搜索文件名为“文档 1”的 Word 文档文件。

```
Sub Searchfile()
    Set fs = Application.FileSearch
    With fs
    .NewSearch
    .FileName = “文档 1.*”
    .FileType = “msoFileTypeWordDocuments”
    .LookIn = “C:\My documents”
```

```
        .SearchSubFolders = False
            If .Execute(SortBy:=msoSortByFileType)>0  Then
                Set newdoc = Documents.Add
                newdoc.Activate
                For i = 1 To .FoundFiles.Count
                        Selection.TypeText Text:=.FoundFiles.Item(i)
                        Selection.TypeParagraph
                Next
            Else
                    MsgBox "No such files found"
            End If
        End with
        newdoc.SaveAs ("Search Results")
    End Sub
```

11.1.5　搜索包含指定内容的文档

在搜索文档时，我们不仅可以根据文件名和文件类型搜索符合条件的文件，还可以搜索（在文档的正文和属性中）包含指定内容的文档。也就是说，可以对文档的内部（正文和属性）进行搜索，找出符合搜索条件的文件。

用 FileSearch 对象的 TextOrProperty 属性可以返回或设置在查找指定文件的过程中要查找的单词或短语，它们可位于一个文件的正文或文件属性中。并且该单词或短语可包含 *（星号）或 ?（问号）通配符。

在搜索包含指定内容的文档时，可以设定查找的方式是精确查找还是模糊查找。

如果要对文档进行精确查找，则应将 FileSearch 对象的 MatchTextExactly 属性设置为 True。此时只有在文档中包含指定单词或短语的完全匹配形式时，文档才被列入到搜索结果之中。

如果要对文档进行模糊查找，则应将 FileSearch 对象的 MatchAllWordForms 属性设置为 True。那么只要在文档中包含指定单词或短语的任意形式，该文档就会被列入到搜索结果之中。例如如果我们将 TextOrProperty 属性的值设为“Go”，并且将 MatchAllWordForms 属性设为 True，那么所有包含单词“Goes”，“Went”，“Gone”或者“Go”的文档都将被列入搜索结果。

在下例中，我们对指定的目录进行搜索，并列出在正文或属性含有 单词“Go”的任何形式的的文件，并将其输出到文件“Search Results”中。

```
Sub Searchfile()
    Set fs = Application.FileSearch
```

```
    With fs
    .NewSearch
    .TextOrProperty = “Go”
    .MatchAllWordForms = True
    .LookIn = “C:\My documents”
    .SearchSubFolders = False
        If .Execute(SortBy:=msoSortByFileType)>0  Then
            Set newdoc = Documents.Add
            newdoc.Activate
            For i = 1 To .FoundFiles.Count
                    Selection.TypeText Text:=.FoundFiles.Item(i)
                    Selection.TypeParagraph
            Next
        Else
                MsgBox "No such files found"
        End If
    End with
    newdoc.SaveAs ("Search Results")
End Sub
```

11.1.6 设置搜索条件

PropertyTests 对象集合代表的是在文件搜索过程中所有的搜索条件。PorpertyTest 对象则是 PropertyTests 对象集合中的一个元素。

用 FileSearch 对象的 PropertyTests 属性可返回一个 PropertyTests 对象。以下示例显示查找单个文件的“查找”的搜索条件数。

```
Application.FileSearch.PropertyTests.Count
```

用 PropertyTests(index) 可返回一个 PropertyTest 对象；此处 index 是该对象的索引号。以下示例用于显示 PropertyTests 集合中第一个搜索条件。

```
Sub DisplayPropertyTest()
  With Application.FileSearch.PropertyTests(1)
  myString = "This is the search criteria: " _
      & " The name is: " & .Name & ". The condition is: " _
      & .Condition
  If .Value <> "" Then
      myString = myString & ". The value is: " & .Value
      If .SecondValue <> "" Then
          myString = myString _
              & ". The second value is: " _
```

```
                    & .SecondValue & ", and the connector is" _
                    & .Connector
                End If
            End If
            MsgBox myString
            End With
        End Sub
```

✍ 在上例中，PorpertyTest 对象的 Name 属性和 Condition 属性分别用于返回搜索条件的类型和具体的搜索条件。

用 PropertyTests 集合的 Add 方法可以向 PropertyTests 集合中添加一个新的 PropertyTest 对象。

Add 方法的语法格式如下：

expression.Add(Name, Condition, Value, SecondValue, Connector)

上式中各项的意义分别如下：

expression：代表搜索条件集合，整个表达式返回一个 PropertyTests 对象。

Name：String 类型，必需。代表搜索条件的类型。例如：如果搜索条件是查找所有在正文或属性中含有单词“go”的文件。那么这个搜索条件的 Name 属性值应为：“Text or Property”

Condition：Variant 类型，必需。代表具体的搜索条件。该参数值

可设置为以下 MsoCondition 常量之一：

Value：Variant 类型，可选。测定搜索条件的值。

SecondValue：Variant 类型，可选。查找范围的上限。只有当 Condition 取 msoConditionAnyTimeBetween 或 msoConditionAnyNumberBetween 时，才可使用此参数。

Connector：Variant 类型，可选。指定两个搜索条件的组合方式。该参数值可设置为以下 MsoConnector 常量之一：msoConnectorAnd（和） 或 msoConnectorOr（或）。

以下示例向搜索条件中添加搜索条件。第一个条件指明查找任意类型的文件，第二个条件指明该文件应是在 1996 年 1 月 1 日至 1996 年 6 月 30 日之间修改的。然后，在消息框中显示找到的文件数和文件名。

```
Sub AddPropertyTest()
  Set fs = Application.FileSearch
  fs.NewSearch
    With fs.PropertyTests
        .Add Name:="Files of Type", _
            Condition:=msoConditionFileTypeAllFiles, _
            Connector:=msoConnectorOr
        .Add Name:="Last Modified", _
```

```
            Condition:=msoConditionAnytimeBetween, _
            Value:="1/1/96", SecondValue:="6/1/96", _
            Connector:=msoConnectorAnd
    End With
        If fs.Execute() > 0 Then
            MsgBox "There were " & fs.FoundFiles.Count & _
                " file(s) found."
             For i = 1 To fs.FoundFiles.Count
                MsgBox fs.FoundFiles(i)
            Next i
        Else
            MsgBox "There were no files found."
        End If
End Sub
```

使用PropertyTests对象集合的Remove方法可以在搜索条件集合中删除某一个搜索条件（PropertyTest 对象）。

Remove 方法的语法如下：

expression.Remove(lndex)

上式中各项的意义分别如下：

expression：必需。代表搜索条件集合。整个表达式返回一个

PorpertyTests 对象集合。

Lndex：Long 类型，必需。代表要删除的属性测试的索引号。

下面的语句可实现的功能为：删除搜索条件集合中的第 2 个搜索条件。

```
Application.FileSearch.PropertyTests.Remove(2)
```

11.1.7　处理搜索结果

FoundFiles 对象代表由文件查找过程返回的文件列表。每执行一次文件搜索，FoundFiles 对象就得到一次更新。用 FileSearch 对象的 FoundFiles 属性可返回 FoundFiles 对象。

尽管 FoundFiles 是一个对象，但是它有一些类似于集合的属性和方法。例如用它的 Count 属性可以返回文件列表的数目，还可以使用 FoundFiles(index)来返回文件列表中的某一个文件文件名和路径。

在下例可实现：逐个查看找到的文件列表中的文件并显示其中每个文件的文件名和路径。用 FoundFiles(index) 可返回查找过程中指定文件的名字和位置，此处的 index 是该文件的索引号。

```
Sub ShowFiles()
  With Application.FileSearch
      For i = 1 To .FoundFiles.Count
          MsgBox .FoundFiles(i)
```

```
        Next I
    End With
End Sub
```

在执行搜索时，可以定义搜索结果的排序方法。搜索后的结果存放在 FoundFiles 对象之中。通过对 FoundFiles 对象的控制，我们可以很容易地对搜索的结果进行处理。

11.2　用 VBA 操作文件

VBA 提供了内置的文件函数，利用这些函数可以直接操作文件（当然，也包括对 Word 2000 文档进行操作）。例如查看文件名，查看及设置文件属性，建立、复制、移动、删除文件和目录等等。本章将对这些内容进行介绍。

11.2.1　Dir 函数

可以说 Dir 函数是 VBA 中最常用的文件函数。

调用 Dir 函数时，以对文件名的某种描述（例如完整的文件名、文件名的首字符等等）作为参数，Dir 函数将返回符合该描述的文件名。其语法为：

Dir（filename[，attribute]）

✍ 其中的可选参数 attribute 为文件的属性，例如只读、隐藏

等，指定文件属性后，Dir 函数将查找具有该属性的文件。稍后将介绍文件属性的详细知识以及与文件属性有关的 VBA 函数。

调用 Dir 函数时一般应当指定文件的绝对路径。如果不指定路径，则 Dir 函数将默认对当前目录进行搜索。

Dir 函数有多种调用方法（即 Dir 函数语法中的 filename 可有多种形式）。以下将分别进行介绍。

1. 以完整文件名作参数

以完整的文件名作参数时，如果文件存在，则 Dir 函数返回该文件名；如果文件不存在，则 Dir 函数返回空字符串。

示例：

```
Dir（“C:\My Document\letter.doc”）
```

当在指定路径 C:\My Document\下存在文档 letter.doc 时，函数返回“letter.doc”，否则函数返回空字符串。

利用这种方法可以非常方便地检查某个文档是否存在。例如：

```
Dim Check As Boolean
If(Dir(“C:\My Document\letter.doc”)<>“”)Then
   Check=True
Else
   Check=False
```

End If

当路径 C:\My Document\下存在文档 letter.doc 时，Check 的值为 Ture；否则 Check 的值为 False。

上例也可用如下形式：

Dim Check As Boolean

Check=(Len（Dir（“C:\My Document\letter.doc”））>0)

当路径 C:\My Document\下存在文档 letter.doc 时，Len 函数的返回值为 10，此时 Check 的值为 Ture；若指定路径下不存在文档 letter.doc，则 Len 函数的返回值为 0，此时 Check 的值为 False。

2. 以带有通配符的文件名作参数

以带有通配符的文件名作参数时，Dir 函数返回符合要求的第一个文件名。

假设 C:\My Document\下有“letter.doc”、“letter.html”、“let.doc”、“let.html”、“late.doc”、“lend.doc”、“load.doc”以及“s*.*”多个文件。

Dir（“C:\My Document\le*.doc”）　　函数返回“let.doc”

Dir（“C:\My Document*.html”）　　函数返回“let.html”

Dir（“C:\My Document*”）　　函数返回“late.doc”

Dir（“C:\My Document\l??d.*”）　　函数返回“lend.doc”

不带任何参数调用 Dir 函数时，函数将返回符合要求的下一个文件

名。利用这一点可以查找到指定目录下所有符合要求的文件：

```
Sub PrintFileName（）
  Dim Filename As String

  Filename=Dir(“*.*”)
  Do Until Filename=“”
    Debug.Print Filename
     ‘不带参数调用 Dir 函数
   Filename=Dir
  Loop
End Sub
```

程序 PrintFileName 将打印出当前目录下的所有文件名。

3. 以目录名作参数

以目录名作参数时，Dir 函数返回指定目录下第一个文件的文件名。

示例：

假设 C:\My Document\下有“letter.doc”、“letter.html”、“let.doc”、“let.html”、“late.doc”、“lend.doc”、“load.doc”以及“s*.*”多个文件。

Dir（“C:\My Document”）　　　　函数返回“late.doc”

- 第一次调用 Dir 函数时必须指定文件或者目录的路径名；同理，如果 Dir 函数已经返回空字符串，则下一次调用时必须指定路径，否则 VBA 将产生运行期错误。

11.2.2 操作文件属性

每个文件都有自己的属性，例如存档、隐藏、只读、系统、目录等。不同属性的文件，其显示方式或可以对其进行的操作各不相同。例如，可以在 Windows 资源管理器里设置不显示隐藏文件；具有目录属性的文件实际上是一个目录或子目录。又如，假设一篇文档为只读文件，则不能对其进行修改、删除及重命名等操作。由此可知，用户必须对文件属性有所了解，才能对其进行正确的操作。

VBA 可以对文件属性进行操作，包括查看文件属性，修改文件属性等等。以下将分别进行介绍。

1. VBA 中的文件属性常量

对于任何一种文件属性，都对应着不同的值，VBA 中定义了不同的常量来表示这些值。

VBA 文件属性常量如表 11-4 所示。

表 11-4 VBA 文件属性常量

文件属性	值	VBA 常量
缺省	0	Visual BasicNormal
只读	1	Visual BasicReadOnly

隐藏	2	Visual BasicHidden
系统	4	Visual BasicSystem
卷标	8	Visual BasicVolume
目录	16	Visual BasicDirectory
存档	32	VBArchive

✍ 对于某个文件来说，它可能同时具有多种文件属性。例如，某个文件既是隐藏文件，又是只读文件。

2. 查看文件属性

利用函数 GetAttr 可以查看文件属性。其语法为：

GetAttr（Filename）

例如：

GetAttr（“C:\My Document\letter.doc”）

GetAttr（“C:\My Document”）

GetAttr 函数将返回一个包含文件属性长整型值。

✍ 调用 GetAttr 函数时一般应当指定文件的绝对路径。如果不指定路径，则 GetAttr 函数将默认对当前目录进行搜索。

以下为利用 GetAttr 函数编写出的一个查看文件属性的通用程序：

```
Sub FileAttribute(Filename As String)
    Dim FileInt As Integer
    Dim FileStr As String
```

```
    FileInt=GetAttr(Filename)
    FileStr=Filename
    FileStr=FileStr&“的文件属性：”
    If(FileInt And Visual BasicNormal) Then-
                 FileStr=FileStr&“缺省/”
    If(FileInt And Visual BasicReadOnly) Then-
                 FileStr=FileStr&“只读/”
    If(FileInt And Visual BasicHidden) Then-
                FileStr=FileStr&“隐藏/”
    If(FileInt And Visual BasicSystem) Then-
                FileStr=FileStr&“系统/”
    If(FileInt And Visual BasicVolume) Then-
                FileStr=FileStr&“卷标/”
    If(FileInt And Visual BasicDirectory) Then-
                FileStr=FileStr&“目录/”
    If(FileInt And VBArchive) Then-
                FileStr=FileStr&“存档/”
    MsgBox FileStr
  End Sub
```

过程 FileAttribute 可以给出一个对话框，其中显示了要查看的文件

的属性。对于某个指定的文件，要查看其文件属性时，只需直接调用过程 FileAttribute 即可。

例如，要查看目录 My Document 以及该目录下的文档 letter.doc 的文件属性，则可用以下程序实现：

```
Sub ShowAttribute(Filename As String)
  FileAttribuet Filename=“C:\My Document”
  FileAttribuet Filename=“C:\My Document\letter.doc”
End Sub
```

3. 设置文件属性

有时用户需要更改文件的某种属性，或者新增某种文件属性。例如，为了保护某个文档的内容不被改动，可以将其设为只读文件。又如，某个只读文件从别处下载而来，现在想要对其重新命名，则必须去除其只读属性。对文件属性的设置可以利用 VBA 中的 SetAttr 函数来实现。

SetAttr 函数的语法为：

```
SetAttr  filename，attribute[+attribute…]
```

例如：

```
SetAttr  “C:\My Document\letter.doc”，Visual BasicReadOnly
SetAttr  “C:\My Document\letter.doc”，Visual BasicReadOnly+Visual BasicHidden
```

SetAttr　“C:\My Document\letter.doc”，-

GetAttr(“C:\My Document\letter.doc”)+Visual BasicHidden

需要强调的是，在最后一个例句中，必须事先确信文件 C:\My Document\letter.doc 不具有隐藏属性，否则“GetAttr(“C:\My Document\letter.doc”)+Visual BasicHidden”的结果将不是预想的文件属性。

- 利用 SetAttr 函数设置文件属性时，文件原有的属性将被覆盖掉。
- 不能企图用 SetAttr 函数为文件设置目录属性或卷标属性，否则 VBA 将产生运行期错误。

11.2.3　查看文件的其他信息

利用 Dir 函数和 GetAttr 函数分别可以查看文件名和文件属性。除此之外，VBA 还提供了一些函数用于查看文件的其他信息。

1. 查看文件长度

利用函数 FileLen 可以查看文件的长度（字节数）。

FileLen 函数的语法为：

FileLen（Filename）

例如：

GetAttr（“C:\My Document\letter.doc”）

调用 FileLen 函数时，参数 Filename 中不能包括通配符。VBA 不支持类似于以下的一些用法：

GetAttr（“C:\My Document*.doc”）

GetAttr（“C:\My Document\le??.*”）

2. 查看文件日期和时间

利用函数 FileDateTime 可以查看最近一次修改文件的日期和时间。

FileDateTime 函数的语法为：

FileDateTime（Filename）

例如：

GetDateTime（“C:\My Document\letter.doc”）

11.2.4 管理文件与目录

利用 VBA 可以直接对文件和目录进行管理，包括复制、删除、移动、重命名文件，新建、删除目录等。以下将分别进行介绍。

1. 复制文件

利用函数 FileCopy 可以复制文件。

FileCopy 函数的语法为：

FileCopy　FilepathA，FilepathB

例如，要将文档 letter.doc 从目录 C:\My Document\下复制到 D 盘根目录下，可通过如下语句实现：

FileCopy “C:\My Document\letter.doc”,“D:\letter.doc”

如果要将文档letter.doc从目录C:\My Document\下复制到D盘根目录下并且重命名为story.doc，可通过如下语句实现：

FileCopy “C:\My Document\letter.doc”,“D:\story.doc”

也可利用重命名文件的函数Name来实现。关于Name函数将在稍后介绍。

- FileCopy 函数不能复制正打开的文件。例如，如果正用Word 2000对文档letter.doc进行编辑，则此时不能复制该文档。
- 使用 FileCopy 函数复制文件时，如果目标文件（即FileCopy函数语法中的FilepathB）已经存在，则原有文件将被覆盖。因此使用FileCopy函数时应当特别小心。

  VBA不支持这样的用法:

  FileCopy “C:\My Document\letter.doc”, “C:\My Document\letter.doc”

  即不能用FileCopy函数复制文件本身。

2. 删除文件

利用函数Kill可以删除文件。

Kill函数的语法为：

Kill　Filename

例如，要删除目录 C:\My Document\下的文档 letter.doc，可通过如下语句实现：

Kill “C:\My Document\letter.doc”

调用 Kill 函数时，参数 Filename 中可以包括通配符。例如：

Kill “C:\My Document*.doc”

Kill “C:\My Document\le??.*”

Kill “C:\My Document*”

Kill 函数将删除所有符合描述的文件。

- Kill 函数不能删除正打开的文件。例如，如果正用 Word 2000 对文档 letter.doc 进行编辑，则此时不能删除该文档。
- 不能用 Kill 函数删除具有只读、隐藏或者系统属性的文件。

3. 重命名或移动文件

利用函数 Name 可以重命名文件以及移动文件。

利用 Name 函数重命名文件的语法为：

Name FilenameA As FilenameB

例如，要将目录 C:\My Document\下的文档 letter.doc 重命名为 story.doc，可通过如下语句实现：

Name “C:\My Document\letter.doc” As “C:\My Document\story.doc”

利用 Name 函数移动文件的语法为：

Name　FilepathA As FilepathB

例如，要将文档 letter.doc 从目录 C:\My Document\下移动到 C 盘根目录下，可通过如下语句实现：

Name “C:\My Document\letter.doc” As “C:\letter.doc”

如果要将文档 letter.doc 从目录 C:\My Document\下移动到 C 盘根目录下并且重命名为 story.doc，可通过如下语句实现：

‘移动文档

Name “C:\My Document\letter.doc” As “C:\letter.doc”

‘重命名文档

Name “C:\letter.doc” As “C:\story.doc”

也可用如下语句在移动文档的同时重命名文档：

Name “C:\My Document\letter.doc” As “C:\story.doc”

- 不能企图用 Name 函数将文件从一个盘移动到另一个盘。即 FilepathA 和 FilepathB 的驱动器名必须相同。例如下面的语句是错误的：

  Name “C:\My Document\letter.doc” As “D:\letter.doc”

- Name 函数不能重命名或移动正打开的文件。例如，如果正用 Word 2000 对文档 letter.doc 进行编辑，则此时不能重命名或移动该文档。

- 使用 Name 函数重命名或移动文件时，如果目标文件（即 Name 函数语法中的 FilenameB 或 FilepathB）已经存在，VBA 将出现运行期错误。

以上介绍的 FileCopy、Kill 以及 Name 函数是可以结合起来使用的。这将使对文件的管理变得更加灵活。

例如，要将文档 letter.doc 从目录 C:\My Document\下移动到 D 盘根目录下并且重命名为 story.doc，可通过多种过程来实现。

过程 A：

```
Sub MethodA（）
   FileCopy “C:\My Document\letter.doc”,“D:\letter.doc”
   Kill “C:\My Document\letter.doc”
   Name “D:\letter.doc” As “D:\story.doc”
End Sub
```

过程 B：

```
Sub MethodB（）
   FileCopy “C:\My Document\letter.doc”,“D:\story.doc”
   Kill “C:\My Document\letter.doc”
End Sub
```

过程 C：

```
Sub MethodC（）
```

```
    Name "C:\My Document\letter.doc" As-
                    "C:\My Document\story.doc"
    FileCopy "C:\My Document\story.doc","D:\story.doc"
    Kill "C:\My Document\story.doc"
End Sub
```

可以看出，对于本例而言，过程 C 是最简单的一种方法。

注意下面的方法是错误的：

```
Sub WrongMethod（）
    Name "C:\My Document\letter.doc" As "D:\story.doc"
End Sub
```

4. 新建目录

利用函数 MkDir 可以新建目录。

MkDir 函数的语法为：

```
MkDir  Directoryname
```

例如，要在 C 盘新建目录 Document，可通过如下语句实现：

```
MkDir "C:\Document"
```

又如，要在目录 C:\My Document 下新建子目录 Letter，则可通过如下语句实现：

```
MkDir "C:\My Document\Letter"
```

✍ 利用 MkDir 函数新建目录时，如果省略驱动器名，则 VBA

将默认在当前盘下新建目录。

- 利用 MkDir 函数新建的目录名不能与已有的目录名重名。也不能在不存在的目录下新建子目录。

5. 删除目录

利用函数 RmDir 可以删除目录。

RmDir 函数的语法为：

RmDir　Directoryname

例如，要删除 C 盘的目录 Document，可通过如下语句实现：

RmDir　“C:\Document”

又如，要删除目录 C:\My Document 下的子目录 Letter，则可通过如下语句实现：

RmDir　“C:\My Document\Letter”

- 利用 RmDir 函数删除目录时，如果省略驱动器名，则 VBA 将默认删除当前盘下的目录。
- 不能用 RmDir 函数删除当前目录。

需要特别强调的是，利用 RmDir 函数删除目录时，要删除的目录必须是全空的而不能包括任何文件或子目录。如果想要删除某个包含文件及子目录的目录，可以先用 Kill 函数删除该目录及其各级子目录下的所有文件，然后用 RmDir 函数从下往上逐级删除各个空的子目录直至删除要删除的目录。

下面的例子给出了删除包含文件及子目录的目录的一种方法。函数利用循环语句以及对自身的递归调用对各级目录进行操作（首先删除该目录下的所有文件，然后删除该目录），最终将传递给函数的目录内容全部删除。

```
Function　DelDirectory（Directoryname As String）As Boolean
    Dim DDDirectory As String
    Dim Filename As String
    ‘CurDir 函数用于取得当前目录，DDDirectory 暂时存放当前目录名

    DDDirectory=CurDir
    ‘ChDir 函数用于改变当前目录

    ChDir Directoryname
     ‘删除当前目录下所有文件

    Filename=Dir(“*”)
    Do Until Filename=“”
        Kill Filename
        Filename=Dir
```

```
Loop
 ‘建立子目录清单

Do
   Filename=Dir(“*”,Visual BasicDirectory)
    ‘不考虑清单中的“.”和“..”值

   Do While Filename=“.”Or Filename=“..”
      Filename=Dir
   Loop
   If Filename=“”Then
      Exit Do
   Else
       ‘如果目录中还有文件，则递归调用函数以删除

      If Not DelDirectory(Filename)Then
         Go to ExitHere
      End If
   End If
Loop
```

```
‘ 当前目录复位

ChDir DDDirectory
‘删除空目录

RmDir Directoryname
DelDirectory=Ture
ExitHere:
    Exit Function
End Sub
```

第十二章　Word 与其他应用程序的交互引用

12.1　OLE 对象模型

OLE 对象模型是 Microsoft 的一项技术，它允许在一个应用程序中使用另外一个应用程序的命令和功能，允许通过引用另一个应用程序的对象、属性和方法来返回、编辑和输出数据。OLE 自动化的工作方式是：通讯被动方（OLE 服务器）应用程序向通讯主动方（OLE 客户机）应用程序提供一个以上可供其调用的 OLE 自动化对象类型，OLE 客户机通过引用这些对象实现对 OLE 服务器的调用，然后通过设置对象的属性和使用对象的方法操纵 OLE 服务器应用程序，完成两者之间的通讯。Word 2000 是一个支持 OLE 自动化的应用软件，它可以作为 OLE 服务器供其他应用程序调用。

可以由另一个应用程序引用的应用程序对象叫作 Automation 对象。

并不是所有的 Windows 应用程序都可以成为 Automation 对象，也就是说，并不是所有的应用程序都支持自动化管理器。只有那些专门设计的，可由对象模块识别的属性、方法和时间等来获取特殊功能的应用程序才支持自动化管理器。基于 Windows 的可开放其对象的应用

程序被称为服务器程序，使用这些对象的程序被称为客户应用程序。

通过 OLE 自动化，使我们在开发新的应用程序时可以“借用”现成的应用程序的部分或全部功能，从而大大地减轻开发的工作量，缩短开发周期，使开发工作事半功倍。这就是 OLE 自动化带给开发人员的好处。

✍ Microsoft Office 系列程序都支持自动化管理器。

下面我们将介绍两个函数：CreateObject 和 GetObject。通过这两个函数，我们可以实现对支持 OLE 对象模型的应用程序的引用。

1. CreateObject 函数

CreateObject 函数创建并返回一个对应用程序对象的引用。

CreateObject 函数的语法如下：

CreateObject（class，[servername]）

CreateObject 函数的语法各部分的意义如下：

Class：　必需，代表要创建的应用程序名称和类。

class 参数使用 appname.objecttype 这种语法，包括以下部分：

表 12-1　Class 参数的组成

部分	描述
appname	必需的；提供该对象的应用程序名。
objecttype	必需的；Variant（字符串）。待创建对象的类型或类。

每个支持自动化的应用程序都至少提供一种对象类型。例如，一

个字处理应用程序可能会提供 Application 对象，Document 对象，以及 Toolbar 对象。

Servername：可选的；代表要在其上创建对象的网络服务器名称。如果 servername 是一个空字符串(“”)，即代表使用本地机器。

下面的代码将创建对一个 Microsoft Excel 电子数据表的应用，并将其辅给对象变量 ExcelSheet。

```
Dim ExcelSheet As Object
Set ExcelSheet = CreateObject("Excel.Sheet")
```

对象创建后，就可以在代码中使用自定义的对象变量来引用该对象。在下面的示例中，可以使用对象变量 ExcelSheet 来访问新建对象的属性和方法，以及访问 Microsoft Excel 的其他对象，包括应用程序对象和单元格集合。

```
设置 Application 对象使 Excel 可见
ExcelSheet.Application.Visible = True
'在表格的第一个单元中写些文本
ExcelSheet.Application.Cells(1, 1).Value = "This is column A, row 1"
'将该表格保存到 C:\test.xls 目录
ExcelSheet.SaveAs "C:\TEST.XLS"
'使用应用程序对象的 Quit 方法关闭 Excel。
ExcelSheet.Application.Quit
```

'释放该对象变量

Set ExcelSheet = Nothing

使用 As Object 子句声明对象变量，可以创建一个能包含任何类型对象引用的变量。不过，该变量访问对象是后期绑定的，也就是说，绑定在程序运行时才进行。要创建一个使用前期绑定方式的对象变量，也就是说，在程序编译时就完成绑定，则对象变量在声明时应指定类ID。例如，可以声明并创建下列 Microsoft Excel 引用：

```
Dim xlApp As Excel.Application
Dim xlBook As Excel.Workbook
Dim xlSheet As Excel.WorkSheet
Set xlApp = CreateObject("Excel.Application")
Set xlBook = xlApp.Workbooks.Add
Set xlSheet = xlBook.Worksheets(1)
```

可以在一个远端连网的计算机上创建一个对象，方法是把计算机的名称传递给 CreateObject 的 servername 参数。这个名称与共享名称的机器名部份相同：对于一个共享名称为"\\MyServer\Public," 的 servername 参数是 "MyServer" 。

下面的例子返回在一个名为 MyServer 的远端计算机上运行的 Excel 实例的版本号：

```
Dim xlApp As Object
```

```
Set xlApp = CreateObject("Excel.Application", "MyServer")
Debug.Print xlApp.Version
```

如果远端服务器不存在或者不可用，则会发生一个运行时错误。

2. GetObject 函数

GetObject 函数返回对一个应用程序对象的引用。

GetObject 函数的语法如下：

GetObject（[pathname] [, class]）

GetObject 函数的语法包含下面几个命名参数：

Pathname：可选的，包含待检索对象的文件的全路径和名称。如果省略 pathname，则 class 是必需的。

Class：可选的，代表该对象的类的字符串。

其中，class 参数的语法格式为 appname.objecttype，且语法的各个部分如下：

表 12-2 Class 参数的组成

部分	描述
appname	必需的；Variant (String)。提供该对象的应用程序名称。
objecttype	必需的；Variant (String)。待创建对象的类型或类。

使用 GetObject 函数可以访问支持 OLE 对象模型的应用程序对象，而且可以将该对象赋给对象变量。

使用 Set 语句可以将 GetObject 返回的对象赋给对象变量。例如：

Dim DocObject As Object

Set DocObject = GetObject("C:\My documents\mydoc.doc")

当执行上述代码时，就会启动与指定的 pathname 相关联的应用程序，同时激活指定文件中的对象。

如果 pathname 是一个零长度的字符串 ("")，则 GetObject 返回指定类型的新的对象实例。如果省略了 pathname 参数，则 GetObject 返回指定类型的当前活动的对象。如果当前没有指定类型的对象，就会出错。

如果对象已注册为单个实例的对象，则不管执行多少次 CreateObject，都只能创建该对象的一个实例。若使用单个实例对象，当使用零长度字符串 ("") 语法调用时，GetObject 总是返回同一个实例，而若省略 pathname 参数，就会出错。

下面的示例使用 GetObject 函数来获取对指定的 Microsoft Excel 的工作表 (MyXL) 的引用。它使用工作表的 Application 属性来显示或关闭 Microsoft Excel 等等。DetectExcel Sub 过程通过调用两个 API 函数，来查找 Microsoft Excel。如果 Microsoft Excel 正在运行，则将其放入运行对象表（Running Object Table）中。如果 Microsoft Excel 不在运行，则第一次调用 GetObject 将导致错误。在本例中，出现该错误则把 ExcelWasNotRunning 标志设为 True。第二次调用

GetObject 是指定要打开的一个文件。如果 Microsoft Excel 不在运行，则这个第二次的调用将启动该程序，并返回一个指定文件(mytest.xls) 所对应的工作表的引用。该文件必须位于指定的位置；否则将产生 Visual Basic 错误及自动化错误。随后的示例代码将 Microsoft Excel 及包含指定工作表的窗口设为可见。最后，如果在此前没有 Microsoft Excel 的副本在运行，代码就使用 Application 对象的 Quit 方法来关闭 Microsoft Excel。如果该应用程序原来就在运行，则不要试图关闭它。引用本身在设为 Nothing 后被释放。

```
'声明必要的 API 例程：
Declare Function FindWindow Lib "user32" Alias _
"FindWindowA" (ByVal lpClassName as String, _
                    ByVal lpWindowName As Long) As Long

Declare Function SendMessage Lib "user32" Alias _
"SendMessageA" (ByVal hWnd as Long,ByVal wMsg as Long _
                    ByVal wParam as Long, _
                    ByVal lParam As Long) As Long

Sub GetExcel()
    Dim MyXL As Object     '用于存放
```

'Microsoft Excel 引用的变量。

Dim ExcelWasNotRunning As Boolean '用于最后释放的标记。

'测试 Microsoft Excel 的副本是否在运行。

On Error Resume Next '延迟错误捕获。

'不带第一个参数调用 Getobject 函数将

'返回对该应用程序的实例的引用。

'如果该应用程序不在运行，则会产生错误。

Set MyXL = Getobject(, "Excel.Application")

If Err.Number <> 0 Then ExcelWasNotRunning = True

Err.Clear '如果发生错误则要清除 Err 对象。

'检测 Microsoft Excel。如果 Microsoft Excel 在运行，

'则将其加入运行对象表。

DetectExcel

'将对象变量设为对要看的文件的引用。

Set MyXL = Getobject("c:\Visual Basic4\MYTEST.XLS")

```
'设置其 Application 属性，显示 Microsoft Excel。
'然后使用 MyXL 对象引用的 Windows 集合
'显示包含该文件的实际窗口。
    MyXL.Application.Visible = True
    MyXL.Parent.Windows(1).Visible = True
    '在此处对文件
    '进行操作。
    ' ...
'如果在启动时，Microsoft Excel 的这份副本不在运行中，
'则使用 Application 属性的 Quit 方法来关闭它。
'注意，当试图退出 Microsoft Excel 时，
'标题栏会闪烁，并显示一条消息
'询问是否保存所加载的文件。
    If ExcelWasNotRunning = True Then
        MyXL.Application.Quit
    End IF

    Set MyXL = Nothing    '释放对该应用程序
                          '和电子数据表的引用。
```

```
End Sub

Sub DetectExcel()
'该过程检测并登记正在运行的 Excel。
    Const WM_USER = 1024
    Dim hWnd As Long
'如果 Excel 在运行，则该 API 调用将返回其句柄。
    hWnd = FindWindow("XLMAIN", 0)
    If hWnd = 0 Then        '0 表示没有 Excel 在运行。
        Exit Sub
    Else
    'Excel 在运行，因此可以使用 SendMessage API
    '函数将其放入运行对象表。
        SendMessage hWnd, WM_USER + 18, 0, 0
    End If
End Sub
```

12.2　从其他应用程序中自动进行 Word 操作

要使另一个应用程序可以使用 Word 的自动功能，首先应创建一个对 Word 应用程序（Application）对象的引用。

在 VBA 中，可用 CreateObject 或 GetObject 函数返回一个 Word 的 Application 对象。

例如，在 Microsoft Excel 过程中，可使用以下指令启动 Word 程序：

```
Set wrd = CreateObject("Word.Application")
```

该指令使 Word 中的 Application 对象可用于自动功能。通过使用 Word 的 Application 对象的对象、属性和方法，可以在别的程序中控制 Word。例如，下面的语句创建一个新的 Word 文档，输入一行字符，然后保存并关闭该文档。

```
Set wrd = CreateObject("Word.Application")
wrd.Documents.Add FileName:=MyDoc.doc
Selection.TypeText Text:= “Word Doc Open in Excel”
Activedocument.Save
Activedoment.Close
```

使用 Quit 方法可关闭调用的 Word 应用程序。下面的 Microsoft Excel 示例显示 Word 的启动路径。在显示启动路径之后，用 Quit 方法来关闭刚显示的 Word。

```
Set wrd = CreateObject("Word.Application")
MsgBox wrd.Options.DefaultFilePath(wdStartupPath)
wrd.Quit
```

✍ 在其他程序中还可以调用 Word 2000 中的宏，其方法如下:

将 Word 2000 应用程序赋给一个对象变量。

```
Dim myapp as Object
Set myapp=CreateObject("Word.Application")
```

创建一个新文档

```
myapp.Documents.Add
```

假设读者编写了一个名为“Micro14()”的宏，则可以用下面的语句运行之

```
myapp.Run "Micro14"
```

关闭应用程序，并释放对象变量

```
myapp.Quit
Set myapp=nothing
```

12.3　在 Word 中调用其他应用程序

要在 Word 中通过自动化管理器调用其他应用程序，首先需使用 CreateObject 或 GetObject 函数获得应用程序的引用。然后，使用另一个应用程序的对象、属性和方法进行信息的添加、更改或删除操作。在完成更改之后，再关闭应用程序。

下面的 Word 示例显示 Microsoft Excel 的启动路径。可以使用 Set 语句和 Nothing 关键字来清除一个对象变量，效果等同于关闭该应用程序。

```
Set myobject = CreateObject("Excel.Application")
MsgBox myobject.StartupPath
Set myobject = Nothing
```

可用 OLE 编程标识符创建一个 Automation 对象，然后用这个对象控制应用程序或者控件。在表 12-3~12-9 中列出了主要的 AxtiveX 控件、Microsoft Office 应用程序的标示符。

表 12-3 ActiveX 控件的标示符

ActiveX 控件	**标识符**
复选框	Forms.CheckBox.1
组合框	Forms.ComboBox.1
命令按钮	Forms.CommandButton.1
框架	Forms.Frame.1
图像	Forms.Image.1
标签	Forms.Label.1
列表框	Forms.ListBox.1
多页控件	Forms.MultiPage.1
选项按钮	Forms.OptionButton.1
滚动条	Forms.ScrollBar.1
数值调节钮	Forms.SpinButton.1
TabStrip	Forms.TabStrip.1
文本框	Forms.TextBox.1
切换按钮	Forms.ToggleButton.1

要 Microsoft Office 应用程序创建为一个 Automation 对象，可使用

下表中列出的标识符。

表 12-4　Word 2000 对象的 OLE 标示符

Word 2000 对象	标识符
Application	Word.Application；Word.Application.9
Document	Word.Document；Word.Document.9, Word.Template.8
Global	Word.Global

表 12-5　Access 对象的标示符

Access 对象	标识符
Application	Access.Application；Access.Application.9
CurrentData	Access.CodeData；Access.CurrentData
CurrentProject	Access.CodeProject；Access.CurrentProject
DefaultWebOptions	Access.DefaultWebOptions

表 12-6　Excel 对象的标示符

Excel 对象	标识符	注释
Application	Excel.Application；Excel.Application.9	
Workbook	Excel.AddIn	
Workbook	Excel.Chart；Excel.Chart.8	返回包含两个工作表的工作簿；一个用于图表，一个用于数据。图表工作表是活动工作表。
Workbook	Excel.Sheet；Excel.Sheet.8	返回带一个工作表的工作簿。

表 12-7　Microsoft Graph 对象的标示符

对象	标识符

Application	MSGraph.Application；MSGraph.Application.8
Chart	MSGraph.Chart；MSGraph.Chart.8

表 12-8　Microsoft Outlook 对象的标示符

Outlook 对象	标识符
Application	Outlook.Application；Outlook.Application.9

表 12-9　Microsoft PowerPoint 对象的标示符

Power Point 对象	标识符
Application	PowerPoint.Application；PowerPoint.Application.9

第十三章　防范宏病毒知识介绍

使用 VBA 编辑宏，可以给我们的工作带来巨大的便利；但是任何事物都有其两面性，由于 VBA 语言易于使用并且功能强大，它也成为病毒撰写者偏爱的一种语言。实际上，无论是从数量上还是从“质量”上，Word 宏病毒都超过其他类型的病毒。还记得前两年让人望而生畏的“美丽莎”病毒吗？本章将介绍宏病毒的机理及防毒方法。

13.1　宏病毒的有关知识

13.1　什么是宏病毒

宏病毒是一种寄存在文档或模板的宏中的计算机病毒。一旦打开这样的文档，其中的宏就会被执行，于是宏病毒就会被激活，转移到计算机上，并驻留在 Normal 模板上。从此以后，所有自动保存的文档都会“感染”上这种宏病毒，而且如果其他用户打开了感染病毒的文档，宏病毒又会转移到他的计算机上。

如果某个文档中包含了宏病毒，我们称此文档感染了宏病毒；如果 WORD 系统中的模板包含了宏病毒，我们称 WORD 系统感染了宏病毒。

虽然 Word 2000 自身无法扫描软盘、硬盘或网络驱动器上的宏病毒（要得到这种保护，需要购买和安装专门的防病毒软件。)。但当打开一个含有可能携带病毒的宏的文档时，它能够显示宏警告信息。这样就可选择打开文档时是否要包含宏,如果希望文档包含要用到的宏(例如，单位所用的定货窗体)，打开文档时就包含宏。如果您并不希望在文档中包含宏，或者不了解文档的确切来源。例如，文档是作为电子邮件的附件收到的，或是来自网络或不安全的 Internet 节点。在这种情况下，为了防止可能发生的病毒传染，打开文档过程中出现
宏警告提示时最好选择“取消宏”。

13.2　宏病毒的分类

根据触发条件，Word 宏病毒可以分为两类：

1．公用宏病毒

这类宏病毒在启动或调用 Word 文档时自动触发执行，对所有的 Word 文档有效。

这类宏病毒有两个显著的特点：

（1）宏病毒的名称是用“Auto”开头。也就是说，这类宏病毒只能用“Autoxxxx”来命名。其中，xxxx 表示一种宏文件名，如 AutoOpen、AutoClose 等；

（2）附加在 Word 共用模板上。它们都是附加在 Normal.dot 模板

上。

2．私用宏病毒

这种宏病毒与前者的主要区别在于，它一般是依附在用户自定义的某个特定 Word 模板中，只有用户打开使用这个特定模板的文档时，该宏病毒才有效，而打开使用其它模板的文档时，该宏病毒一般不起作用。

下面介绍两种典型的宏病毒：

1．Nuclear 宏病毒：

这是一个对操作系统文件和打印输出有破坏功能的宏病毒。这个宏病毒中包含以下病毒宏：

AutoExec

AutoOpen

DropSuriv

FileExit

FilePrint

FilePrintDefault

FileSaveAs

InsertPayload

Payload

这些宏是只执行（Execute-only）宏。

Nuclear 宏病毒造成的破坏现象为：

（1）打开一个染毒文档并打印的时侯，它会在您打印的最后一段加上“STOPALLFRENCHNUCLEARTESTINGINTHEPACIFIC!”，这个现象是在每分钟的 55 秒—60 秒之间操作打印时发生。

（2）如果在每天 17：00—18：00 之间打开一个染毒文档，Nuclear 病毒会将 PH33R 病毒传染到计算机上，这是个驻留型病毒。

（3）在每年的四月五日，该病毒会将计算机上 IO.SYS，

MSDOS.SYS 文件清零，并且删除 C 盘根目录上的 COMMAND.COM 文件。一旦病毒发作，MS－DOS 就不可能被引导，计算机将陷入瘫痪。

2．台湾一号病毒

台湾一号病毒会在每月的 13 日影响您正常使用 Word 文档和编辑器。它包含以下病毒宏：

AutoClose

AutoNew

AutoOpen

这些宏是可被编辑宏。

在病毒宏中含有如下的语句：

IfDay(Now())=13Then...

这条语句与 13 日有关。

台湾一号病毒运行在中文 Word 上，其发作时间为每月 13 号，用户打开文件时出一道四则运算题，往往这道题必须经过本机的 VBA 运算才能作对，而用计算器或其它机器均有可能产生错误；如果答对，则进行宏病毒的问答，如“我是宏病毒，如何预防我”，答案是“不要看我”。如果有一个地方答错，则创建 20 个文档，并不断截取硬盘和内存空间，直至系统崩溃。

13.3　宏病毒的特点

1．传播极快

Word 宏病毒通过 DOC 文档及 DOT 模板进行自我复制及传播，而计算机文档是交流最广的文件类型。多年来，人们大多重视保护自己计算机的引导部分和可执行文件不被病毒感染，而对外来的文档文件基本是直接浏览使用，这给 Word 宏病毒传播带来很多便利。特别是 Internet 网络的普及，E-mail 的大量应用更为 Word 宏病毒传播铺平道路。

2．制作、变种方便

Word 使用宏语言 VBA 来编写宏指令。宏病毒同样用 VBA 来编写。目前，世界上的宏病毒原型已有几十种，其变种与日骤增，追究其原因还是 Word 的开放性所致。现在的 Word 病毒都是用 VBA 语言所写成，大部分 Word 病毒宏并没有使用 Word 提供的 Execute—Only 处理

函数处理，它们仍处于可打开阅读修改状态。所有用户在 Word 工具的宏菜单中很方便就可以看到这种宏病毒的全部面目。当然会有“不法之徒”利用掌握的 Basic 语句把其中病毒激活条件和破坏条件加以改变，立即就生产出了一种新的宏病毒，甚至比原病毒的危害更加严重。

3．破坏可能性极大

鉴于宏病毒用 VBA 语言编写，VBA 语言提供了许多系统级底层调用，如直接使用 DOS 系统命令，调用 WindowsAPI，调用 DDE、DLL 等。这些操作均可能对系统直接构成威胁，而 Word 在指令安全性完整性上检测能力很弱，破坏系统的指令很容易被执行。宏病毒 Nuclear 就是破坏操作系统的典型一例。

宏病毒都有如下共性：

1．宏病毒会感染 DOC 文档文件和 DOT 模板文件。被它感染的 DOC 文档属性必然会被改为模板而不是文档，而用户在另存文档时，就无法将该文档转换为任何其它方式，而只能用模板方式存盘。这一点在多种文本编辑器需转换文档时是绝对不允许的。

2．病毒宏的传染通常是 Word 在打开一个带宏病毒的文档或模板时，激活了病毒宏，病毒宏将自身复制至 Word 的通用（Normal）模板中，以后在打开或关闭文件时病毒宏就会把病毒复制到该文件中。

3．大多数宏病毒中含有 AutoOpen，AutoClose，AutoNew 和 AutoExit 等自动宏。只有这样，宏病毒才能获得文档（模板）操作控制权。有些宏病毒还通过 FileNew，FileOpen，FileSave，FileSaveAs，FileExit 等宏控制文件的操作。

4．病毒宏中必然含有对文档读写操作的宏指令。

13.4　宏病毒的传播机理

一般来说，一个宏病毒传播发生在被感染的宏指令覆盖、改写及增加全局宏指令表
中的宏，由此进一步感染随后打开和存贮的所有 Doc 文档。

当 Word 打开一个.doc 文件时，先检查里面有没有模板/宏代码，如果有的话就认为这不是普通的 doc 文件，而是一个模版文件，并执行里面的 auto 类的宏(如果有的话)。一般染毒后的.doc 被打开后，通过 Auto 宏或菜单、快捷键和工具栏里的特洛伊木马来激活，随后感染诸如 Normal.dot 或 powerup.dot 等全局模板文件得到系统"永久"控制权。夺权后，当系统有文档存储动作时，病毒就把自身复制入此文档并储存成一个后缀为.doc 的模板文件；另外，当一定条件满足时，病毒就会干些小小的或者大大的破坏活动。

由于宏病毒利用了 Word 的文档机制进行传播，它和以往的病毒防治方法不同，一般情况下，人们大多注意可执行文件（.com exe）的

病毒感染情况，而 Word 宏病毒寄生于 Word 的文档中，而且人们一般都要对文档文件进行备份，因此病毒可以隐藏很长一段时期。

目前，国内发现的宏病毒主要在大公司和外企单位中传播。由于该病毒能跨越多种平台，并且针对数据文档进行破坏，因此具有极大的危害性，宏病毒在公司通过内联网相互进行文档传送时，迅速漫延，往往不到一周的时间就能使公司的机器全部染上病毒。

13.2 Word 2000 的防毒特性

Word 2000 提供了下述的防病毒特性：

1．Word 2000 的安全设置。

2．支持第三方防病毒软件。

3．Microsoft 代码认证技术允许开发人员使用数字证书对他们的解决方案中的 VBA 工程进行数字签名。

Word 2000 的安全设置

Word、Excel、PowerPoint 和 Outlook 中的安全设置，类似于 Internet Explorer 中可以使用的安全设置。Office 2000 提供了三种安全设置，它们会影响包含宏的文档：

- High（高）：只允许那些由可信任证书签名过的宏自动运行。所有未签名的宏不能运行且不进行通知。如果宏是由不在受信

源列表中出现的签名签署的，那么系统就显示一个对话框。这时用户所能做的工作，则取决于管理员所作的选择：

如果系统管理员没有锁定受信源列表，那么用户在打开一个签过名、但以前还不是可信任文档时，就可以有选择地向受信源列表添加当前这个签名。

如果系统管理员在添加可信任的签名后已经锁定了受信源列表，用户就不能再向受信源列表添加新的签名了，并且只有使用现有可信任证书进行签名的宏才能运行。

如果系统管理员没有添加任何可信任的签名就锁定了受信源列表，任何宏都不能运行。

- Medium（中等）：只允许那些使用可信任证书签名过的宏自动运行。当新的签名添加到受信源列表中时，前面对 High（高）设置所说明的那些管理限制同样在这里适用。但是在 Medium（中等）设置中，用户可以根据需要选择让宏运行，而不管它们是否被签过名。
- Low（低）：用户不会收到任何警告：打开所有文档，所有宏都可以运行。这种设置仅当用户安装了防病毒软件并且不想使用数字签名时才推荐使用。

1. 设置安全级别

要设置安全设置值，那么，将鼠标指向“工具”菜单的“宏”，然

后单击“安全性”命令，将显示“安全性”对话框，如图 14-1 所示。

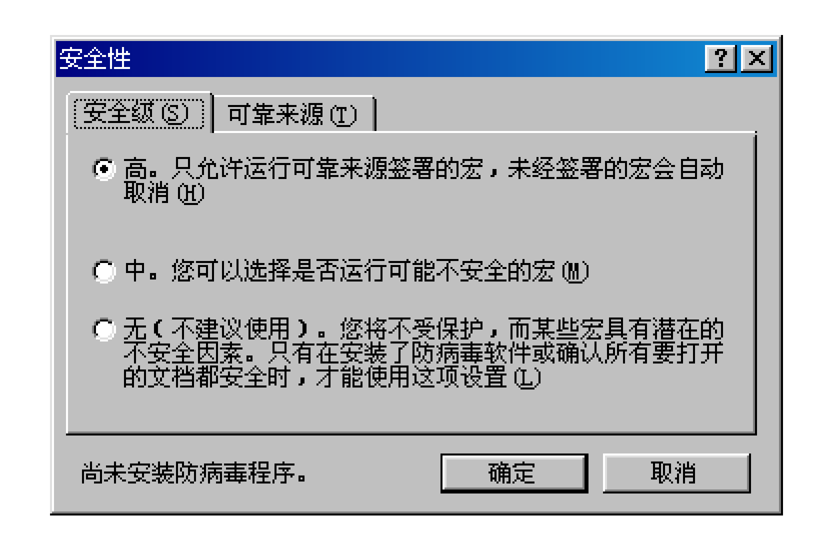


图 13-1　“安全性”对话框

下表摘要总结了 Security Level（安全级别）选项卡中各种设置、不同的文档类型和签名验证结果对宏病毒防护的影响。在所有设置中，如果安装了防病毒软件并且文档中包含了宏，那么在文档打开和验证之前对文档检查已知病毒。

表 13-1　文档的安全设置对宏病毒防护的影响

文档类型和验证结果	**高（安全性）**	**中**	**低**
没有宏	打开文档	打开文档	打开文档
未签名的宏	关闭宏的自动运行且不进行任何通知，打开文档或 Outlook 应用程序	提示用户开放或关闭宏的运行	无提示，开放宏
受信源签名的	自动开放宏，打开文	自动开放宏，打开文	无提示

宏，验证成功	档或 Outlook 应用程序	档或 Outlook 应用程序	或验证，开放宏
未知作者签名的宏，验证成功	显示对话框，给出证书信息。只有通过在 Security Warning（安全警告）对话框中选中 Alwaystrust macros from thisauthor（总是信任来自这个作者的宏）复选框，用户选择信任这个作者和验证授权时，宏才能运行。网络管理员可以锁定受信源列表，防止用户通过向该列表添加作者而使文档或 Outlook VBA 工程的宏能运行	显示对话框，给出证书信息。提示用户选择开放或关闭宏的运行。下述任务可根据需要选择：通过在 Security Warning（安全警告）对话框中选中 Always trustmacros from this author（总是信任来自这个作者的宏）复选框，用户可以选择信任该作者和验证授权	无提示或验证，开放宏
来自任意作者签名的宏，验证失败，可能是由于病毒的影响	警告用户可能存在病毒，自动关闭宏	警告用户可能存在病毒，自动关闭宏	无提示或验证，开放宏

续　表

文档类型和验证结果	**高（安全性）**	**中**	**低**
来自任意作者签名的宏，验证不能完成，原因是找不到公开密钥或证书使用了不兼	警告用户不能进行验证，自动关闭宏	警告用户不能进行验证，提示用户开放或关闭宏	无提示或验证，开放宏

容的加密算法			
来自任意作者签名的宏，签名是在证书失效或作废后进行的	警告用户签名已过期或作废，自动关闭宏	警告用户签名已过期或作废，提示用户开放或关闭宏	无提示或验证，开放宏

2。使用第三方防病毒软件

Word、Excel 和 PowerPoint 现在支持一个新的应用程序编程接口（API），允许第三方防病毒软件在打开文档时，对文档及其所基于的模板和安装的加载项进行查杀病毒的工作。有关支持这个新 API 的防病毒软件的更详细信息，参见 Microsoft Office 防病毒中心 Web 站点：http://Officeupdate.microsoft.com/Articles/antivirus.htm。

3。使用数字证书生成可信任的解决方案

Word 2000 使用了 Microsoft 代码认证技术，允许开发人员使用被开发人员标识为受信源的数字证书，为他们的文档、模板和加载项中的 VBA 工程进行数字签名。为了响应这种特性，Word 2000 中拥有类似于 Microsoft Internet Explorer 中的安全级别设置，允许用户识别受信源生成的代码，并确保签署过的代码没有被篡改过。为了生成可信任的解决方案，需要获得并安装数字证书，可以使用它对 Word、Excel、PowerPoint 和 Outlook 解决方案中的 VBA 工程进行数字签名，以确保他们是你所做的工作。当打开一个签署过的文档时，Microsoft Office 应用程序能够对数字签名进行验证，以确定签署过的 VBA 工程是否

被进行过某种改变，并自动地使可能由宏病毒修改过的宏失效。

- 数字签名仅能用于安装了 Internet Explorer 4.0 及以上版本计算机上的 Office 安装中。在安装了 Internet Explorer 3.x 或以前版本、Netscape Navigator 的所有版本或其它浏览器的所有计算机上，都不能使用 Microsoft Office 中的数字签名和签名验证特性。如果用户试图在这样的计算机上对 VBA 工程进行数字签名，系统就会显示一条消息，通知用户由于没有安装 InternetExplorer 4.0 及以上版本而不能使用这一特性。当在这样的计算机上打开包含宏（不论签署与否）的任意文档时，Office 应用程序就会显示一个标准的宏病毒对话框，它允许用户在打开文档之前选择打开宏或关闭宏。

下述几段将讨论数字签名的概念和技术，并讨论在 Word 2000 中如何使用这一技术来识别可信任的解决方案。

（1）什么是数字证书

可以把数字证书看作是 ID 卡（身份卡）（比如驾驶员的驾驶执照或护照）的电子对应物。数字证书的原理与发行物理 ID 卡的系统相似。你向称之为证书授权机构（Certification Authority）的可信任公共团体（比如是 VeriSign,Inc 或 ThawteConsulting.）提供自己的信息，证书授权机构证实你的信息后就向你签发数字证书。数字证书包含了该证书发给了谁以及发出该证书的证书授权机构的信息。另外，一些

证书授权机构自身可能由一个或多个证书授权机构以层次结构方式来认证，这些信息也是证书的一部分。当数字证书用于对文档和软件进行签名时，这个 ID 信息和签署项以安全且可验证的形式存储起来，这样它就能够向用户表达可信任的关系。

数字证书使用被称为“公开密钥密码”的加密技术，签署软件出版物并验证证书本身的完整性。公开密钥密码使用一对匹配的加密和解密密钥，分别被称为公开密钥和私有密钥。公开密钥密码算法对数据实施单向变换，所以用一个密钥加密的数据，只能用与之配对的另一个密钥来解密。另外，每个密钥都使用一个相当大的数值，使得从一个公开密钥推导相对应的私有密钥在计算上成为不可能。正因为如此，一个公开密钥可以广泛使用而不会带来安全上的危险。

为了进一步降低某人从一个公开密钥推导其私有密钥的可能性，认证机构对密钥对加上时间戳，使得必须周期地更换它们，并提供另外的机制确保在证书失效之前才能用于签名。在数字证书的有效期内使用的签名，将无限期地有效（除非签署项被篡改或签名被删除）。数字证书失效后进行的签名是无效的。

为了理解公开密钥密码的工作方式，首先说明如何使用它加密电子邮件消息将会很有益处。为了使用公开密钥密码方法进行加密，发方首先从目录服务中获得收方的公开密钥，然后在发送消息之前先把消息加密。当收方收到消息后，收方使用自己的私有密钥对消息脱密。

只要私有密钥保持了安全，其它用户就不能对消息进行脱密，收方就确保了消息在传输过程中没有被篡改。

一种类似的系统用于对文档和软件进行数字签名。它不是加密整个文件，而是对文件执行一个单向散列算法，生成一个称之为消息摘要（Message Digest）的东西。消息摘要是一个可以看作是文件的“数字指纹”的唯一数值。Microsoft Office 2000 使用 MD5 散列算法生成消息摘要。生成消息摘要增加了创建及以后修改大型文档和软件文件签名过程的有效性。然后，使用签名者的私有密钥对消息摘要进行加密，生成附加在文件之上的数字签名。

为了验证文件的完整性，打开文件的应用程序首先使用同一个散列函数生成文件的消息摘要，然后使用签名者的公开密钥对附加在文件之上的签名进行脱密，恢复文件原来签署时生成的消息摘要，比较这两个消息摘要，如果文件的任意部分被修改或破坏，那么这两个消息摘要就不会相同，文件的内容就是不可信任的。这时，验证过程就会失败，不管文件是如何被修改的棗是通过破坏、宏病毒，还是由加载项或 Office 解决方案以可编程方式进行的改变。如果文件没有使用有效证书进行签署，验证过程也会失败，这里所说的失效证书指的是，证书已经失效，或者是伪造的、变更的或破坏的证书。如果另一个用户修改了 VBA 工程，Office 应用程序就删除当前签名，并提示用户对 VBA 工程进行重新签名。如果用户不签署 VBA 工程，或使用另一个

证书进行签署，该文件的验证过程就会失败。

- 一方面，前面的说明描述了用于对文档和软件两者都能进行的数字签名的过程，另一方面，Word 2000 应用程序却只提供了对包含于文档中的 VBA 工程进行签名的能力。Word 2000 应用程序不提供对文档本身进行签名以确保文本和其它内容（除了宏）不发生改变。

同一个数字签名过程也用于数字证书本身，用以识别数字证书是不是由认证机构生成的，确保证书没有被篡改或伪造。

为了进一步提高安全性，数字证书都有由认证机构强加的有效期。典型情况下，数字证书从发出之日起一年之内有效。这样做的目的是为了大大降低恶意人员从公开密钥推导出其对应的私有密钥的可能性。尽管从公开密钥推导出对应的私有密钥是极不可能的，因为计算大的密钥值所需的可能组合的数量十分巨大。但是从理论上来说并非没有可能。另外，添加有效期限制了证书被盗后的存活期，证书的有效期确保了证书失效后所做的任何签名都无效。

如果发出证书的证书授权机构提供了“时间戳服务”，那么就将待签署的散列代码通过 Internet 或内联网连接发送到证书授权机构维护的时间戳服务器上进行验证。如果代码是在签名有效存活期中签署的，就向签名添加一个时间戳，那么即使是在证书失效后签名仍然有效。如果分发数字证书的证书授权机构不支持时间戳，那么使用证书所作

的所有签名在证书失效后也就变为无效的了。

（2）创建自己的数字证书

当把读者的 Office 2000 应用程序的安全级别设置成 High（高）时，只有由受信源签署过的宏才可以运行；在所有其它文档中的宏都不能运行。如果打算对自己编写的宏使用 High（高）安全级别设置，那么，或者如本节后面说明的那样，需要从证书授权机构获得证书，或者创建你自己使用的数字证书。

要创建自己使用的数字证书，请运行创建数字证书（Create DigitalCertificate）实用程序（Selfcert.exe），并把自己的信息输入进去，这些信息将保存在证书中。由于这样创建的数字证书不是由正式的证书授权机构发出，所以这样创建的数字证书被称之为“自签署证书”，使用这样的证书签署的 VBA 工程被称为“自签署工程”。

创建自签署证书前，必须首先安装创建数字证书实用程序(Selfcert.exe)，如果在安装 Office 2000 过程中选择 Typical（典型安装），则不会自动安装这个实用程序。

安装创建数字证书实用程序的步骤为：

1．在控制面板中双击 Add/Remove Programs（添加/删除程序）。

2．在 Install/Uninstall（安装/卸载）选项卡中，单击“Microsoft Office2000”，然后单击 Add/Remove（添加/删除）。

3．在 Microsoft Office 2000 Maintenance Mode（Microsoft Office

2000 维护方式）对话框中单击 Add or Remove Features（添加或删除特性）按钮。

4．展开 Office Tools（Office 工具），将 Digital Signature for VBA Projects（VBA 工程数字签名）设置为 Run from My Computer（从我的电脑中运行）。

5．单击 Update Now（现在更新）。Selfcert.exe 将与 Office 2000 应用程序安装在同一个文件夹中，缺省文件夹是 C:\Program Files\Microsoft Office\Office。

创建自签署数字证书的步骤为：

1．在“我的电脑”中或 Windows 资源管理器中运行 Selfcert.exe。

2．在 Your Name（你的名称）框中，输入自己的名字和你想保存在该证书中的任何其它识别信息，然后单击 OK（确定）。Selfcert.exe 将创建并安装自签署证书，你可以使用它对当前计算机上的 VBA 工程进行签名。为了创建在另一个计算机上使用的自签署证书，就要在那个计算机上运行 Selfcert.exe 程序。

✍ 在大多数情况下，由 Selfcert.exe 创建的自签署证书，仅限于个人使用或测试目的。为了以最安全的模式使用 Microsoft Office 宏病毒保护特性，你和你的组织应该只使用由证书授权机构发出的证书对 VBA 工程进行签名。如果使用自签署证书对 VBA 工程进行签名，那么当安全级别设置成 Medium

（中）或 High（高）后，当第一次打开包含已签署过 VBA 工程的文档时，就会显示一个 Security Warning（安全警告）对话框，指出用于签署 VBA 工程的证书并非由证书授权机构发出，不应该信任它。显然，你可以安全地信任由你自己使用自签署证书签名的 VBA 工程。这样，当第二次打开文档时，就不会再次显示 Security Warning（安全警告）对话框了。但是，作为一项一般性的安全策略，一个组织应该要么锁定受信源列表，防止用户信任那些不是由管理员提供的证书，要么坚决劝阻用户信任那些由自签署证书签名的 VBA 工程。

（3）备份数字证书或向另一个计算机传送数字证书

如果安装了 Microsoft Internet Explorer 5，就能够备份数字证书或向另一个计算机传送数字证书。要实现这一功能，请使用 Certification Manager(证书管理器)移出或移入你的证书：

使用 Certification Manager（证书管理器）移出或移入证书的步骤为：

1．右击桌面上的 Internet Explorer 图标，然后再单击快捷菜单中的 Properties（属性）。

2．在 Contents（目录）选项卡上单击 Certificates（证书）。系统将显示“Certification Manager(证书管理员)”对话框，该对话框列出了计算机上已经安装的全部证书。你的个人证书罗列在 Personal

（个人）选项卡上。

3. 如果要移出证书，那么选择列表中的证书，然后单击 Export（移出），系统启动 Certification Manager Export Wizard(证书管理器移出向导)，按照向导对话框中的指令，把你的证书保存到文件中。

注释　为了使用个人数字证书签署 VBA 工程，数字证书必须包括私有密钥。当移出个人数字证书时，确保所做的选择包括了它的私有密钥。

4．如果要移入证书，那么，单击 Import（输入）。系统启动 CertificationManager Import Wizard（证书管理器输入向导），按照向导对话框中给出的指令，安装已经保存在文件中的证书。

13.3　如何判断文档是否染毒

虽然不是所有包含宏的文档都包含了宏病毒，但当有下列情况之一时，您可以百分之百地断定您的 OFFICE 文档或 OFFICE 系统中有宏病毒：

1．在打开“宏病毒防护功能”的情况下，当您打开一个您自己写的文档时，系统会会弹出相应的警告框。而您清楚您并没有在其中使用宏或并不知道宏到底怎么用，那么您可以完全肯定您的文档已经感染了宏病毒。

2．同样是在打开“宏病毒防护功能”的情况下，您的 OFFICE 文

档中一系列的文件都在打开时给出宏警告。由于在一般情况下我们很少使用到宏， 所以当您看到成串的文档有宏警告时，可以肯定这些文档中有宏病毒。

3．如果软件中关于宏病毒防护选项启用后，不能在下次开机时依然保存。WORD97 中提供了对宏病毒的防护功能，它可以在“工具/选项/常规”中进行设定。但有些宏病毒为了对付 OFFICE97 中提供的宏警告功能，它在感染系统（这通常只有在您关闭了宏病毒防护选项或者出现宏警告后您不留神选取了“启用宏”才有可能）后，会在您每次退出 OFFICE 时自动屏蔽掉宏病毒防护选项。因此您一旦发现您的机器中设置的宏病毒防护功能选项无法在两次启动 WORD 之间保持有效，则您的系统一定已经感染了宏病毒。也就是说一系列 WORD 模板、特别是 normal.dot 已经被感染。

鉴于绝大多数人都不需要或着不会使用“宏”这个功能，我们可以得出一个相当重要的结论：如果您的 OFFICE 文档在打开时，系统给出一个宏病毒警告框，那么您应该对这个文档保持高度警惕，它已被感染的几率极大。

13.4　防毒于未然

从上面的论述中我们知道：病毒通过 Auto 宏和特洛伊木马来激活，通过修改全局模板文件来传播。对此，我们有如下的办法防止自己的

文档染上病毒。

1．禁止 Word 执行 Auto 类的宏

先来“照顾”Auto 宏。如果你不用向导类的模板（其后缀一般为.wiz）的话（我想一般人都用不到的），完全可以禁止 Auto 宏的执行，方法很简单，自己建立一个叫做 Autoexec

的宏，里面写上如下一句就可以了。

DisableAutoMacros 1

此宏是 Auto 宏里的老大哥，仅在 Word 启动时执行一次。即使对它有了修改，只要不重启 Word 就不会起作用的。

2．备份全局模板文件

这步是最后的防线，需要我们格外保护的是 4 个全局模板文件:

\MSOffice\Template\Normal.dot

\MSOffice\Winword\Startup\Powerup.dot

\MSOffice\Winword\Startup\Prcaddin.dot

\MSOffice\Winword\Startup\Symbar.dot

把这几个文件做一个备份。这样，至少在 Word 能保证 Word 每次重启动时是处于无毒状态的。

13.5　杀除宏病毒

13.5.1　用杀毒软件清除宏病毒

使用反病毒软件是一种高效、安全和方便的清除方法，也是一般计算机用户的首选方法。但是宏病毒千变万化，并不象某些厂商或麻痹大意的人那样认为的有所谓“广谱”的查杀软件。

因此，对付宏病毒应该和对付其它种类的病毒一样，也要尽量使用最新版的查杀病毒软件。无论你使用的是何种反病毒软件，及时升级是非常重要的。

13.5.2 手工杀除宏病毒

如果有一个文档文件(假设为 a.doc)被发现染毒了，如果 Word 2000 本身没有被感染的话，手工杀毒的步骤如下：

1．关闭并重启 Word，如果系统说因为只读无法写 normal.dot 或其他 dot 时，千万别上当，打开只读导致让病毒感染了 Normal.dot 等系统模板文件。

2．为染毒文件做个备份

3．在 Word 里“工具”菜单中选“模板”，再点“管理器”按钮，在“宏”栏先点“关闭文件”，使它变成“打开文件”，打开染毒的 a.doc，然后删除里面所有的宏，再把它关闭

4．打开修改过的 a.doc，观察是否有异常

5．如果 a.doc 一切正常，在资源管理器中，右键击 a.doc，然后选新建，再把新建文档存盘即可。

13.5.3 挽救染毒的文档

如果您的 Word 2000 没有感染宏病毒，但需要打开某个外来的、已查出感染有宏病毒的文档，而手头现有的反病毒软件又无法查杀它们，那么您可以试验用下面的方法来查杀文档中的宏病毒：

首先确认将 Word 2000 宏的安全性设置为“高”，然后打开这个包含了宏病毒的文档，然后在“文件”菜单中选择“另存为”，将此文档改存成写字板（RTF）格式或 Word 6.0 格式。在上述方法中，存成写字板格式是利用 RTF 文档格式没有宏，存成 Word 6.0 格式则是利用了 Word 文档在转换成纯文本格式时会失去宏的特点。写字板所用的 rtf 格式适用于文档中的内容限于文字和图片的情况下，如果文档内容中除了文字、图片外还有图形或表格，那么按 WORD6.0 格式保存一般不会失去这些内容。

存盘后应该检查一下文档的完整性，如果文档内容没有任何丢失，并且在重新打开此文档时不再出现宏警告则大功告成。